Universidad Zaragoza

Estudio de caracterizadores visuales para la detección de obstáculos en vídeos de ski con cámara subjetiva

Escuela de Ingeniería y Arquitectura

TRABAJO FIN DE GRADO
GRADO INGENIERÍA INFORMÁTICA
ESPECIALIDAD COMPUTACIÓN

Alumno: Héctor Francia Molinero
Director: Gregorio de Miguel Casado

Fecha: 20 de abril de 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Héctor Francia Molinero,

con nº de DNI 72992124-Z en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Estudio de caracterizadores visuales para la detección de obstáculos en vídeos de ski con cámara subjetiva, (Título del Trabajo)

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 20 de Abril 2016

Fdo:

Resumen

Este trabajo se centra en el estudio de caracterizadores visuales para la identificación de objetos u obstáculos sencillos en vídeos de ski. Para ello se han utilizado técnicas de aprendizaje para desarrollar un prototipo software que se apoya en un conjunto de prueba creado expresamente para este estudio.

A nuestro saber este tipo de técnicas no se habían aplicado antes a este campo, por lo que se ha tenido que crear una base de datos con imágenes tomadas en primera persona. Como resultado del proyecto se ha permitido comprobar que, para determinados caracterizadores, se obtienen buenos resultados llegando incluso al 90 % de precisión en el reconocimiento de las clases de objetos creadas.

La memoria aborda un análisis del estado del arte, donde se resumen una serie de dispositivos de motorización de la actividad física (pulseras, smartwatches, la nube de aplicaciones que proporcionan servicios extendidos a éstos dispositivos...). El estado del arte también resume los principales artículos relacionados con este estudio y sobre los cuales se apoya, tanto en las técnicas de visión basadas en caracterizadores visuales como en las de aprendizaje. A continuación se presenta la arquitectura del sistema, con un resumen global, la descripción de los caracterizadores visuales SURF (Speed Up Robust Feature), HOG (Histogram of Oriented Gradients), HOF (Histogram of Optical Flow) y MBH (Motion Boundary Histogram) utilizados. Seguidamente, el prototipo del sistema presenta de una manera estructurada toda la implementación realizada. Finalmente, se mostrarán los resultados con su correspondiente evaluación y la gestión del proyecto con sus conclusiones.

A mi padre porque me guió parte del camino y a mi madre por llegar hasta el final,
porque sin ellos no sería quien soy ni habría llegado tan lejos.

Agradecimientos[1]

A D. Antonio Francia Mambrona por regalarme la cámara con la que he podido grabar todos mis vídeos y por acompañarme una y otra vez a esquiar.

A D. Gregorio de Miguel Casado, por creer en mí, por su paciencia, saber hacer y constante apoyo.

A D. Jorge Gracia Sancho y Dña. Emmanuelle Dégremont por dejarse grabar y participar en mis vídeos.

A Dña. Miriam Lucia Pérez por apoyarme día a día y hacerme la vida mucho más fácil.

[1] En orden alfabético

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Vivimos en una época de continuo desarrollo tecnológico en la que cada día se innova y se crea nueva tecnología que nos ayuda en nuestra vida cotidiana, haciéndola más fácil y entretenida.

En estos años en los que la computación ubicua se encuentra cada vez más afianzada entre nosotros, en los que la realidad aumentada ha vuelto a la mente de los desarrolladores, y en los que se puede conseguir una notable mejora de la experiencia de usuario con los nuevos avances en realidad virtual, he decidido realizar una pequeña aportación con este trabajo centrándome en la práctica de actividades deportivas.
La monitorización en el deporte está cada vez más de moda con aplicaciones para móviles que registran itinerarios, pulseras y relojes inteligentes que miden desde la frecuencia cardíaca hasta las calorías quemadas, e incluso ropa equipada con sensores que interpreta la información que recoge y nos envía nuestros resultados. Todo este auge de nuevas tecnologías y dispositivos aplicados al deporte me llevó a plantearme que una mezcla entre mi pasión por la nieve y una parte de todo lo que he aprendido durante estos años, era posible.

Tras varios años practicando Snowboard, grabándome mientras lo hacía y con la experiencia adquirida hasta el momento, me pareció una idea bastante interesante aplicar técnicas de visión por computador y aprendizaje automático a ese conjunto de vídeos, para poder reconocer y diferenciar los objetos que aparecen, con el objetivo futuro de poderlo hacer en el mismo momento en el que se practica, implementado sobre hardware especializado o de propósito general, una vez que los dispositivos móviles tengan la potencia computacional requerida.

Para ello, he realizado el siguiente proyecto, donde analizo diferentes métodos para estudiar las características relevantes de las imágenes, de modo que la comparación entre

ellos permitirá proponer la mejor solución. Como soporte para este trabajo se propone un prototipo software que servirá para dar pie a una futura aplicación, la cual pueda llegar a convertirse en una herramienta indispensable para cualquier amante de la nieve.

Objetivos del proyecto y estructura de la memoria

El objetivo general del proyecto es el estudio e implementación de caracterizadores visuales para la identificación de obstáculos sencillos sobre vídeos de snowboard con cámara subjetiva. Los objetivos concretos plantean la búsqueda de técnicas representativas en visión para la caracterización de imágenes en movimiento y el desarrollo de un prototipo software junto con una base de datos de prueba que permita evaluar la adecuación y complejidad de la algoritmia subyacente.

Este objetivo se corresponde directamente con los apartados de la memoria, que son los que siguen:

Capítulo 2.
Se hablará del estado del arte, proyectos relacionados y técnicas utilizadas.

Capítulo 3.
Se mostrará la arquitectura del sistema con una descripción global y sus requisitos.

Capítulo 4.
Se mostrará el prototipo del sistema, su diseño e implementación.

Capítulo 5.
Se hablará de cómo se ha gestionado el proyecto y las conclusiones obtenidas.

Capítulo 2

Estado del arte

En este apartado se describen una serie de dispositivos y aplicaciones relacionadas con lo que se está estudiando. Posteriormente se realiza una revisión de aquellos trabajos que, por tener un fin similar al de este estudio, pudieran servir como base de discusión para desarrollar una solución para el problema a resolver.

2.1. Dispositivos de monitorización de la actividad física

A continuación se van a detallar instrumentos utilizados para medir la actividad física. En ésta línea describimos las pulseras de actividad física, smartwatches con Adroid Wear o Watch OS y otros dispositivos innovadores muy utilizados en el ámbito del deporte. Para terminar se detallarán algunas de las aplicaciones y programas más conocidas y recomendadas por usuarios aficionados al ejercicio.

2.1.1. Pulseras de actividad física

Microsoft Band

Microsoft Band es uno de los mejores dispositivos de fitness que se pueden adquirir ahora mismo, aunque todavía de la sensación de estar en desarrollo. Su pantalla táctil es útil para recibir todo tipo de notificaciones, y es el escaparate perfecto para la plataforma de salud de Microsoft.

Es capaz de recolectar gran cantidad de información y con ello realizarnos tablas de ejercicios para mantenernos en forma. Se puede usar y configurar desde iOS, Windows

Phone y Android, por lo que es una opción multiplataforma en el mundo polarizado entre iOS y Android.

Xiaomi Mi Band

Xiaomi cuenta con la pulsera de actividad física más barata del mercado, pero no por ello falta de utilidad. Llevará la cuenta de los pasos dados, calorías quemadas, distancia recorrida andando y corriendo, y fases de sueño. También vibra para notificar llamadas, despertarte y se puede desbloquear el teléfono con ella.

Puesto que no lleva GPS o monitor de ritmo cardíaco, las cifras en realidad son orientativas en algunos casos (distancia y caloría quemadas por ejemplo). Pero por menos de 20 euros, con una buena aplicación para Android para establecer objetivos de actividad, es una gran inversión.

Fitbit Charge HR

Esta pulsera incluye un detector de ritmo cardíaco que funciona continuamente, y es ideal para tener medidas exactas de las calorías que quemas, los pasos que das, ciclos de sueño o distancias recorridas. Tiene una autonomía en torno a los cinco días.

Jawbone UP2

Este producto de Jawbone es una buena pulsera de actividad, con una buena cantidad de funcionalidades. Su podómetro es muy preciso así como la información que captura durante nuestro sueño, y la inclusión de alarma inteligente para despertarnos con una suave vibración en el momento más oportuno es de gran ayuda para comenzar de la mejor manera posible el día. Quizás la única pega es que es sólo resistente a salpicaduras en vez de a prueba de agua.

Garmin Vívofit

Una pulsera buena y básica, capaz de medir distancias, ciclos de sueño y con resistencia al agua. Las principales pegas es que la pantalla no cuenta con retroiluminación, y tampoco incluye un altímetro o alarma inteligente, por lo que es de características limitadas, pero que cuenta con autonomía para un año y no hace falta estar quitándosela antes de ir a la ducha.

2.1.2. Smartwatches con Android Wear

Huawei Watch

El Huawei Watch trae un diseño elegante y clásico y supone una enorme evolución respecto a su predecesor. Entre las especificaciones de este reloj destacan una pantalla de 400 x 400 píxeles, con la mejor proporción pantalla-cuerpo vista en un smartwatch de Android Wear hasta ahora.

Moto 360 (3ª generación)

La segunda edición del Moto 360 ha supuesto un gran paso respecto a su primera versión. El nuevo smartwatch de Motorola ahora viene en dos variantes: una con una pantalla de 1,37 pulgadas y otra de 1,56 pulgadas. El nuevo modelo ha perfeccionado todos los aspectos positivos del reloj original y ha mejorado los aspectos mediocres del primer Moto 360.

El Moto 360 (2015) cuenta con dos tipos de batería: la edición más grande tiene una capacidad de 400 mAh y la pequeña cuenta con 300 mAh. El resto de especificaciones son similares a las de cualquier otro smartwatch funcionando con Android Wear: 512 MB de RAM, 4 GB de memoria interna y un procesador Snapdragon 400.

ASUS ZenWatch 2

Uno de los smartwatches con mejor relación calidad-precio. El nuevo reloj de ASUS viene con una etiqueta de precio de 149 €. Por este precio tenemos un reloj con una pantalla de 1,45 pulgadas (y otra versión de 1,63) altamente personalizable. Existen 3 colores de esfera y 18 diferentes pulseras.

LG Watch Urbane

Se trata del que prometía ser el mejor smartwatch de Android Wear y, a pesar de que en muchos aspectos lo es, el LG Watch R no ha conseguido que sus especificaciones estén a la altura de su precio.

Su pantalla es de 1,3 pulgadas, P-OLED de 320 x 320 píxeles, mientras que su procesador es un Snapdragon 400 a 1,2 GHz.

Su batería es posiblemente la mejor característica de este terminal que, pese a no ser excelente, cuenta con una de las mejores autonomías vistas en los relojes con Android Wear.

Sony Smartwatch 3

A diferencia de la mayoría de sus competidores, el Sony Smartwatch 3 tiene un diseño discreto y fino, factor que ha sido criticado y alabado a partes iguales.

Su batería es de 420 mAh, que es suficiente para aguantar entre día y medio y dos días, dependiendo del uso. La carga se hace a través de un cable microUSB, así que incluso el propio cargador de nuestro smartphone Android es válido para aumentar la duración de este reloj.

Trae un procesador Quad ARM A7 a 1,2 GHz y 512 MB de RAM, mientras que su memoria interna es de 4 GB. Incluye NFC, Bluetooth 4.0, micrófono y su pantalla es de 1,6 pulgadas y resolución 320 x 320 píxeles, dando como resultado un peso total de 45 gramos.

2.1.3. Smartwatches compatibles con Android

Samsung Gear S2

Samsung ha fabricado el primer smartwatch con marco giratorio. Esta herramienta permite navegar a través del menú del reloj y manejar las diferentes opciones. Además, el diseño es uno de los mejores vistos en un smartwatch, con tres opciones disponibles: S2, S2 3G y S2 Classic.

El reloj funciona con Tizen OS, el sistema operativo propio de Samsung. Sin embargo, es compatible con teléfonos que utilicen la versión Android 4.4 en adelante.

Pebble Time

El Pebble Time ha sido un enorme éxito de ventas debido a su diseño original y funcional. La nueva generación trae una pantalla ePaper a color que mantiene la gran duración de la batería, cercana a los 6 días. El Pebble te notificará de todo lo que necesites y podrás instalar decenas de apps de terceros. Además, cuenta con las Smartstraps, que permiten interactuar con el hardware externo y los sensores.

LG Watch Urbane LTE

Muy similar al Watch Urbane normal, pero con diferencias importantes como el NFC y la posibilidad de realizar pagos con el reloj. Además, incluye conectividad LTE gracias a su ranura para tarjeta microSIM.

El sistema operativo del Watch Urbane LTE es una versión de WebOS, que LG ya utiliza en sus televisores, pero que ha sido adaptado a este dispositivo. En términos de hardware, es igual que el Watch Urbane, pero su batería crece hasta los 700 mAh, y la compañía asegura que llega a 30 horas de duración con cada carga completa.

Fitbit Surge

Fitbit Surge es un reloj conectado interesante. Cuenta con conectividad 3G y localización por GPS, por lo que será muy útil para muchos de los que salen a correr. Le acompaña una buena cantidad de sensores, incluido ritmo cardíaco, para llevar un buen registro de nuestra actividad física, ciclos de sueño y base de datos de alimentos.

La mayor pega que tiene Fitbit Surge es su diseño, ya que siendo un producto de 300 euros se podría haber esperado algo mucho mejor en este aspecto. Es algo más grueso de lo que les gustaría a muchos usuarios, pero tiene una gran cantidad de funcionalidades que consiguen contrarrestar este aspecto.

2.1.4. Smartwatches con Watch OS

Apple Watch

El famoso reloj de Apple te da la posibilidad de mejorar tu actividad física diaria marcándote objetivos tales como estar 1 minuto de pie por cada hora, moverte más para quemar más calorías o acumular 30 minutos al día de actividad moderada. También puedes monitorizar todo el ejercicio que tengas por costumbre hacer seleccionándolo en el reloj y marcándote objetivos de tiempo o calorías que, una vez completados, podrás guardar en la app del Iphone.

2.1.5. Otros Dispositivos

Google Glass

Google Glass es un dispositivo de visualización de realidad aumentada desarrollado por Google. Tiene muchas aplicaciones y una buena parte de ellas se pueden usar para el deporte. Hoy en día ya son utilizadas en muchos campos de fútbol para recoger datos, grabar imágenes de los jugadores, obtener información general del partido que se está jugando...

Recon Snow2

Gafas especialmente pensadas para esquiadores, son capaces de mostrar la velocidad gracias a un receptor GPS integrado, guarda datos minuciosos de cada salto o giro realizado durante el descenso, muestra la posición de amigos y familiares, los mapas de la pista... Conectado por Bluetooth al móvil, Snow 2 almacena los datos de cada sesión de esquí para compararlos luego con el histórico de descensos.

Nike+ Hyperdunk

Zapatillas de baloncesto que incorporan sensores de serie y que nos permiten obtener datos por ejemplo, de la presión realizada sobre las zapatillas, altitud al realizar saltos y velocidad a la que corremos. Toda ésta información es enviada al móvil y puede ser visualizada a través de la aplicación Nike+.

Adidas Smart Ball

Balón de fútbol con numerosos sensores, lo que nos proporciona una gran cantidad de datos como las zonas de golpeo de balón y su rotación, velocidades, potencias de golpeo, etc., los cuales son realmente útiles para mejorar nuestra técnica con el balón.

Éstos datos son enviados a tu móvil donde se pueden visualizar y compartir para retar a tus amigos en la aplicación miCoach de Adidas.

Babolat Play Puré Drive

Babolat, una de las marcas líderes en la fabricación de raquetas, también ha implementado la tecnología wearable en sus productos. Ésta raqueta es capaz de cuantificar todas las variables de cada golpeo que realicemos a la bola de tenis. Por ejemplo, te puede informar del número de servicios, golpes de revés, drives o smashes que has realizado durante tu juego.

Todo ello se puede visualizar en su aplicación móvil, donde puedes compartir tus resultados, entrar en un ranking con todos los usuarios o consultar tus estadísticas.

2.1.6. "Nube" de aplicaciones

Nike+ Running

Aplicación de Nike sólo para correr, que ofrece planes de entrenamiento y competición con tus amigos. Permite realizar un seguimiento de la ruta, la distancia, el ritmo y el tiempo. Integra también un reproductor de música el cual se puede controlar junto con el mapa desde un único lugar.

Adidas miCoach

Aplicación existente para móviles y web que permite personalizar y registrar tus entrenamientos, seguimiento de tus resultados, consulta de consejos útiles... Algunos aspectos destacables son la posibilidad de crear listas de reproducción para tus entrenamientos y poder añadir zapatillas (incluso si no son Adidas) para que registre cuántos kilómetros has hecho con ellas.

El pack Runtastic

Runtastic ofrece una aplicación principal que permite monitorizar una serie de actividades concretas. Pero, simultáneamente, ofrece otras más centradas en deportes concretos (como el ciclismo de montaña, los abdominales, etc.). Todas ellas son de gran calidad y ofrecen un amplio rango de opciones.

RunKeeper

La alternativa a Runtastic. Buena, pero con menos opciones disponibles que Runtastic.

Endomondo

De Endomondo se podría destacar la gran cantidad de deportes que permite monitorizar. En lo demás, permite, con ínfimas diferencias, lo mismo que sus principales rivales.

Strava

Muy de moda entre los usuarios más avanzados. Permite monitorizar de forma detallada nuestras salidas de running y nuestras salidas con la bicicleta, ofreciendo informes detallados de altitud, velocidades, recorridos, etc.

MyFitnessPal

La salud no solo es ejercicio, también es controlar los nutrientes que ingerimos y para ello, MyFitnessPal nos puede ser de gran ayuda. Además se sincroniza con numerosas apps de fitness, por lo que podremos obtener informes más detallados y amoldados a nuestra situación.

Garmin Connect

Garmin Connect es una herramienta de entrenamiento en línea que te permite almacenar, analizar y compartir todas tus actividades físicas. Tiene algunos aspectos interesantes como, en el caso de participar en carreras, te puede mostrar el mapa, la temperatura, las vueltas que llevas... También puedes controlar tu progreso y compartir toda tu actividad como si de una red social se tratara, en la cual puedes ver recorridos de otros compañeros, sus marcas personales o incluso buscar los trayectos creados por otros usuarios de Garmin Connect por si tus sesiones se están volviendo algo rutinarias.

2.2. Técnicas de Visión artificial

La utilización de técnicas y herramientas derivadas del Aprendizaje para el reconocimiento de elementos visuales en entornos con movimiento de cámara, está en el centro de interés de la comunidad científica ya que puede encontrarse trabajos de investigación sobre este tema en contextos muy variados. Por un lado, desde la perspectiva del aprendizaje, destaca la generalización de uso de diccionarios de palabras visuales empleados como base en la técnica que se conoce como "bag of words". Este es el caso de trabajos de investigación como [Liu et al., 2009], [Wang et al., 2013], [Yuan et al., 2010] y [Faraki et al., 2014].

Destaca en este sentido el trabajo de [Faraki et al., 2014] sobre "Log-Euclidean Bag of Words for Human Action Recognition" ya que no solo se pone de manifiesto el potencial de esta técnica sino también su extensión en el dominio de la Geometría Rienmaninana, que proporciona beneficios computacionales derivados de la caracterización del diccionario visual en un espacio transformado. Por otro lado, desde la perspectiva de identificar los descriptores visuales que mejor se adaptan a los distintos conjuntos de pruebas con imágenes que se utilizan en visión, se percibe que el foco de interés de los trabajos de investigación en el tema, no sólo está en probar y mejorar los que ya se manejan en otras áreas de investigación de visión, sino también en la integración de los mismos, bien sea

de manera directa o jerárquica [Zhang et al., 2006], [López-Sastre et al., 2013], [Uijlings et al., 2014].

A continuación se desarrollan algunos aspectos de los trabajos de investigación mencionados en lo que respecta a los objetivos del proyecto.

2.2.1. Estudio sobre características locales y núcleos para la clasificación de texturas y categorías de objetos [Zhang et al., 2006]

En este artículo se ha investigado el rendimiento de un kernel usado para clasificar categorías de texturas y objetos usando descriptores locales de imágenes. Los resultados que obtienen muestran que se han alcanzado altos niveles de rendimiento con representaciones de imágenes que son esencialmente histogramas.

Una de las contribuciones que realiza es una evaluación completa de múltiples detectores de puntos clave, niveles de geometría invariante, descriptores de "features" y kernels clasificadores.

2.2.2. Detección de personas usando histogramas orientados de flujo y apariencia [Dalal et al., 2006]

En este artículo se desarrolla una familia de detectores de alto rendimiento para videos que contienen personas, cámaras y entornos en movimiento. Los detectores combinan descriptores de apariencia basados en el gradiente, con descriptores de movimiento basados en el flujo óptico, juntos en un SVM. Los canales de movimiento y apariencia usan histogramas orientados para alcanzar un descriptor robusto. Se estudian diferentes esquemas de movimiento, pero se llega a la conclusión de que, aunque haya una diferencia considerable de rendimiento entre ellos, cuando se usan individualmente las "features" de movimiento, esas diferencias se reducen en gran medida cuando estas "features" son usadas en combinación de descriptores estáticos de apariencia.

2.2.3. Aprendiendo sobre acciones realistas de personas en películas [Laptev et al., 2008]

Este artículo presenta una aproximación de cómo recopilar automáticamente datos de entrenamiento para acciones humanas, además de demostrar que esos datos pueden ser usados para entrenar un clasificador para reconocimiento de acciones. El procedimiento automático que usan para etiquetar las películas de su dataset alcanza un 60 % de precisión y escala fácilmente para un gran número de tipos de acciones. El método usado para

la clasificación de acciones amplia el método de reconocimiento de imagen al dominio espacio-temporal y logra el mejor rendimiento sobre reconocimiento hasta el momento de su publicación. Además, demuestran que tienen una alta tolerancia al ruido en el conjunto de entrenamiento, por lo que es apropiado para el aprendizaje de acciones en escenarios automáticos. Finalmente demuestran resultados prometedores en el reconocimiento para 8 tipos de acciones en películas. En él se basan muchos otros ya que es el primero que utiliza los descriptores espacio temporales (HOG, HOF) para crear una bolsa de palabras y clasificar con una SVM.

2.2.4. Reconocimiento de acciones realistas sobre vídeos al aire libre [Liu et al., 2009]

Artículo que presenta un framework para reconocer acciones realistas de vídeos al aire libre. Para obtener buenas "features" utilizan señales de movimiento para descartar las peores y también emplean la técnica del "PageRank" para realizar minería sobre "features" estáticas e informativas. Además usan la información teórica basada en el clustering para reconstruir un vocabulario visual semántico.

Sus experimentos verifican que el framework usado es efectivo para reconocer acciones realistas, y usando "features" híbridas, compuestas por estáticas y de movimiento, pueden mejorar la precisión media de reconocimiento. Utiliza detectores como Harris y Hessian.

2.2.5. Reconocimiento de acciones humanas bajo una métrica Logarítmico-Euclideana Riemanniana [Yuan et al., 2010]

En este artículo, se desarrolla un framework para reconocer acciones de bajo nivel en secuencias de vídeo. En dicho framework de reconocimiento, la matriz de covarianza de las "features" de bajo nivel, se usa para representar secuencias de vídeo bajo la métrica Log-Euclidean Riemannian. El descriptor es compacto, distintivo y tiene poco coste de complejidad computacional. Además, se emplea la distancia EMD para medir la diferencia entre los videos en lugar de usar la tradicional distancia Euclidiana de los histogramas.

Los experimentos realizados con dos datasets prueban la efectividad y robustez del framework que proponen.

2.2.6. Trayectorias densas y descriptores de movimiento limitado para reconocimiento de acciones [Wang et al., 2013]

Este artículo introduce una aproximación a la descripción eficiente de video basada en trayectorias densas y descriptores basados en histogramas de movimiento de fronteras. Las trayectorias densas utilizadas demuestran que mejoran anteriores aproximaciones que usaban KLT o correspondencias con descriptores SIFT.

A su vez, los histogramas de movimiento de fronteras, que son calculados a lo largo de las trayectorias densas, muestran excelentes resultados. Éstos son diseñados para ser robustos frente al movimiento de cámara y se ha demostrado que mejoran el estado del arte de los histogramas que producen los descriptores de flujo óptico.

La descripción de los vídeos se ha evaluado extensamente sobre 9 datasets y han demostrado que mejoran significativamente el estado del arte.

Artículo en el que más nos hemos basado donde se utilizan trayectorias densas para ver cómo evolucionan las imágenes dentro de un vídeo. Se usan los descriptores HOG, HOF y MBH para crear un "bag of features" al que luego se le añadirá más información con pirámides espacio temporales.

2.2.7. Histogramas de líneas orientadas para el reconocimiento de la palma de la mano [Jia et al., 2014]

Este artículo investiga cómo mejorar el rendimiento de métodos de aprendizaje sobre el reconocimiento de la palma de la mano. Para conseguirlo, proponen un nuevo descriptor llamado HOL, el cual es robusto frente a escasa iluminación, traslación y varianzas en la rotación.

Como resultado, hasta algunos métodos de aprendizaje simples como PCA y LDA, han logrado rendimientos muy prometedores de reconocimiento sobre algunas conocidas bases de datos.

2.2.8. Bolsa de palabras Log-Euclideana para reconocimiento de acciones [Faraki et al., 2014]

En este artículo se propone una aproximación para ampliar el modelo popular de Bag of Words a un espacio no Euclidiano, el espacio de matrices simétricas definidas positivas (SPD), formadas por descriptores de covarianza de "features" espacio-temporales. Para hacerlo, crean un codebook y posteriormente histogramas, que pueden ser obtenidos

mediante matrices de covarianza, para poder trazar un Bag Of Words Log-Euclideano (LE-BoW), una extensión del convencional BoW usando geometría Riemanniana para matrices SPD.

El ingrediente principal de su propósito es un difeomorfismo que integra funciones Riemannian sobre matrices SPD dentro de un espacio Euclidiano.

Los experimentos que llevan a cabo demuestran que, la aproximación propuesta para clasificación de acciones humanas, mejora el rendimiento propuesto en otros artículos anteriores como el de [Laptev et al., 2008] y [Wang et al., 2013].

2.2.9. Clasificación de vídeos con características HOG/HOF/MBH extraidas dénsamente [Uijlings et al., 2014]

Este artículo presenta una evaluación del término medio entre la eficiencia y precisión computacional para la clasificación de videos usando un Bag Of Words con descriptores como HOG, HOF y MBH.

Proponen diversas mejoras para la velocidad del cálculo de descriptores además de contribuir públicamente con una implementación en Matlab de éstos descriptores.

Capítulo 3

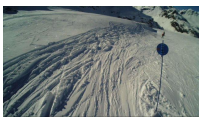
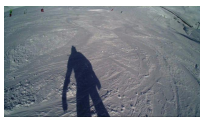
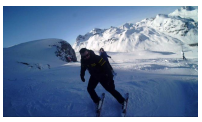
Arquitectura del sistema

En este capítulo de la memoria se presenta la arquitectura general del sistema. En primer lugar se encuentra la descripción del sistema y se detalla cada una de sus partes. A continuación se realiza una descripción de los descriptores utilizados en el estudio y finalmente se enumeran los requisitos necesarios del sistema.

3.1. Descripción global

En la figura 3.1 se describe el proceso general del sistema el cual se basa en la extracción de una cantidad determinada de frames del dataset de videos **(1)**. Las imágenes obtenidas **(2)** se dividen en dos conjuntos, uno de entrenamiento y otro de validación **(3)**. El conjunto de entrenamiento es pasado como parámetro a la función que calculará el Bag Of Words **(4)**. Una vez se tiene el BoW **(5)**, se pasa junto con los datos de entrenamiento a la función que devolverá el clasificador **(6)**. Obtenido el clasificador **(7)**, se inician las pruebas las cuales consisten en clasificar ambos conjuntos **(8)**, para obtener las matrices de confusión que indican cómo de buena ha sido la clasificación **(9)**.

- **Dataset Nieve Videos:** Conjunto de vídeos grabados en la nieve y divididos en las siguientes categorías:

Cañones	Vallas	Balizas	Sombras	Personas
				

Cuadro 3.1: Ejemplos de clases de los datasets

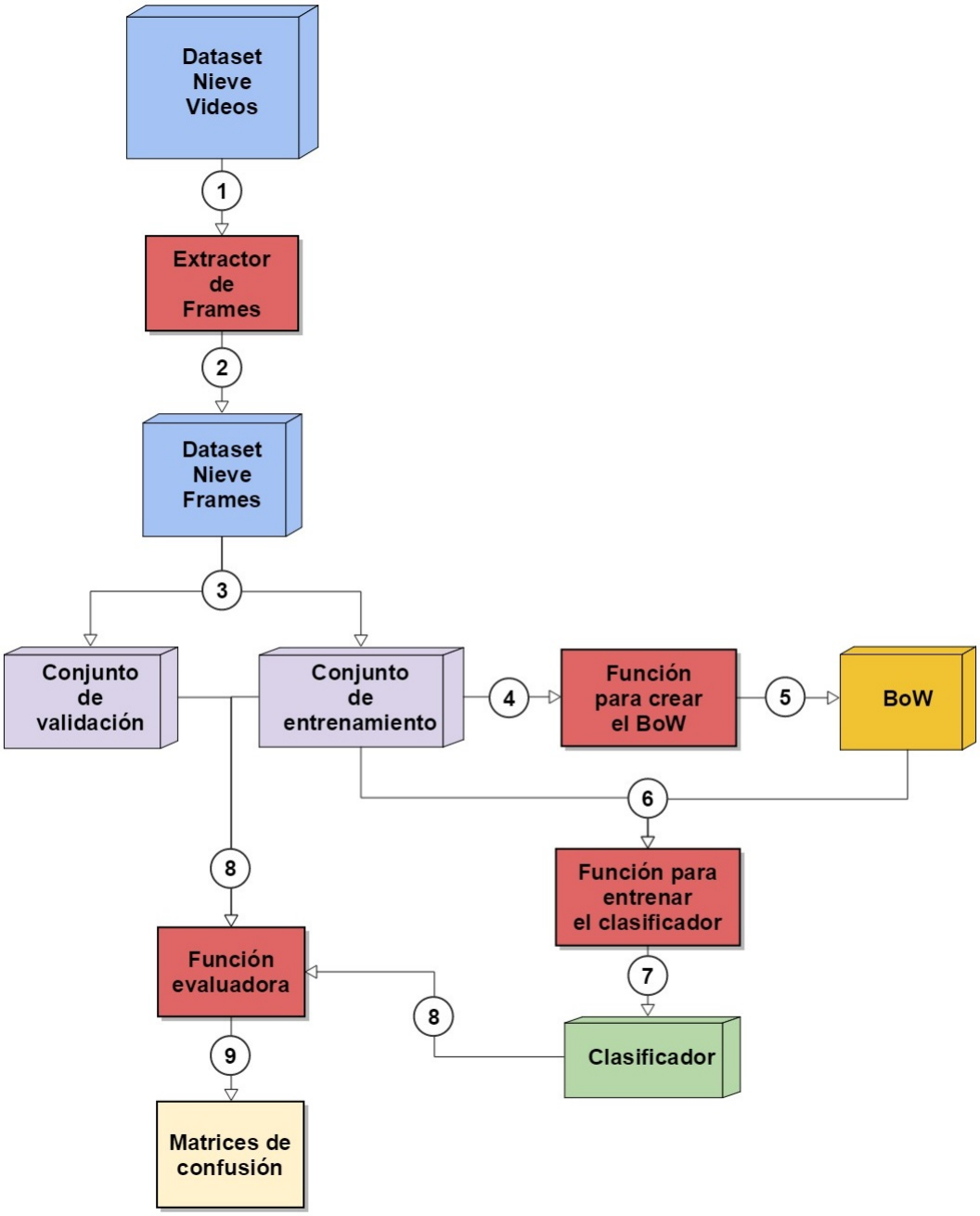


Figura 3.1: Esquema general del proceso con el flujo de operaciones

- **Extractor de Frames:** Función que itera sobre todos los vídeos de cada categoría obteniendo de cada uno un determinado porcentaje de frames.

- **Dataset Nieve Frames:** Conjunto de frames obtenidos de cada vídeo y divididos en las categorías mencionadas anteriormente.

- **Conjunto de entrenamiento:** Contiene el 30 % de los datos del dataset de frames. Este conjunto se utilizará para que el programa aprenda a clasificar las imágenes.
- **Conjunto de validación:** Contiene el 70 % de los datos del dataset de frames. Estos datos no serán utilizados hasta la última fase, donde se evalúa el clasificador.
- **Función para crear el BoW:** Esta función recibe como parámetro el conjunto de entrenamiento junto a una función extractora. El tipo de éste extractor, determina el tipo de descriptor que se obtendrá de cada imagen. Una vez extraídos los descriptores, se construye un vocabulario visual reduciendo el número de features mediante clustering con K-means.
- **Bag of Words (BoW):** Conjunto de características extraídas de ciertos puntos de cada imagen llamadas palabras visuales.
- **Función para entrenar el clasificador:** Las imágenes codificadas de cada categoría sirven de entrada para esta función, la cual utiliza el método de codificación con la bolsa de palabras pasada por parámetro para crear vectores de features que representan a cada categoría.
- **Clasificador:** Algoritmo basado en aprendizaje que permite, dado un conjunto de datos, dividirlos y definir a qué categoría pertenece cada uno.
- **Función evaluadora:** Función que recibe como parámetro el clasificador y ambos conjuntos de datos (entrenamiento y validación), los cuales utiliza para realizar la clasificación y devolver los resultados.
- **Matrices de confusión:** Resultado final basado en una matriz compuesta por valores entre 0 y 1, que describen lo acertada que ha estado la clasificación y los errores que ha habido.

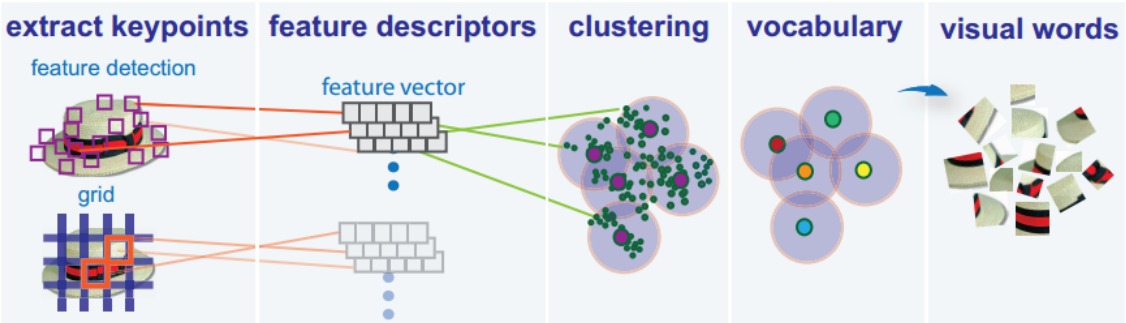


Figura 3.2: Proceso de creación de las palabras visuales [Matlab, 2016].

3.2. Descripción de los descriptores

En esta sección se definen y detallan los descriptores utilizados, resaltando las características propias de cada uno y realizando un resumen de cómo son calculados.

3.2.1. SURF (Speed Up Robust Feature)

Su propósito es la extracción de puntos invariantes en imágenes. Las ventajas principales que brinda el algoritmo son: invariancia ante la escala, orientación y distorsión afín así como invariancia parcial a los cambios de iluminación. El algoritmo se encuentra dividido en 4 etapas principales, las cuales son:

1. Generación de la imagen integral para agilizar los cálculos. Es uno de los principales aportes que utiliza el algoritmo, debido a que contribuyen a una mejora en el funcionamiento. Dada una imagen de entrada y un pixel de esa imagen, la imagen integral es calculada como la suma de los pixeles comprendidos entre el punto y el origen.
2. Uso de un detector basado en la matriz Hessiana. Utiliza el determinante de la matriz para la localización y escala de los puntos, se emplea por su rendimiento en la velocidad de cálculo y precisión.
3. Creación del descriptor. Se trata de un vector de características que se calcula sobre una pequeña región de interés de la imagen, en el caso del descriptor SURF, describe como la intensidad de los pixeles se distribuye dentro de una escala dependiente de la vecindad de cada punto de interés que detectó la Hessiana.
4. Emparejamiento de puntos de interés. Se busca la correspondencia de los puntos de interés identificados en dos imágenes.

3.2.2. HOG (Histogram of Oriented Gradients)

Esta técnica cuenta las ocurrencias de la orientación del gradiente en localizadas porciones de la imagen. Consta de alguna ventaja respecto a otros descriptores dado que, como opera sobre celdas locales, es invariante a transformaciones geométricas. Dadas sus características, es muy adecuado para la detección de personas en imágenes.

Dada una imagen I, los pasos principales para generar HOG son los siguientes:

1. Se divide la imagen completa en $n \times n$ celdas sin solapamiento. Cada celda contiene $c1 \times c2$ pixels.

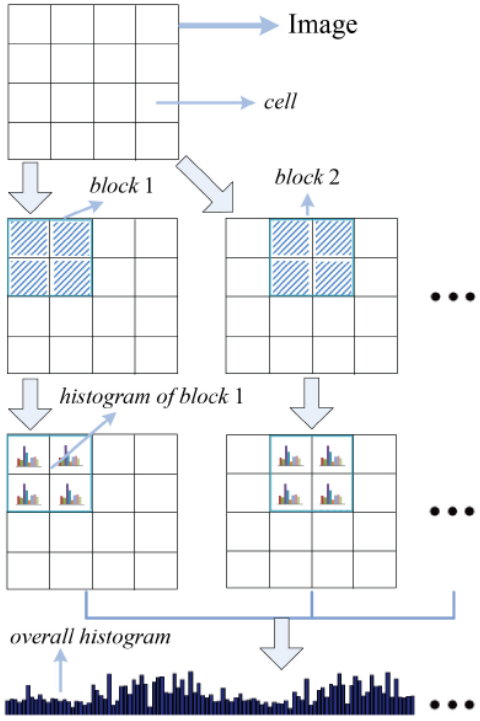


Figura 3.3: Proceso de creación de descriptores HOG [Jia et al., 2014].

2. Se construyen bloques que contienen $b1 \times b2$ celdas. Dos bloques adyacentes se pueden solapar.

3. Para cada pixel $I(x, y)$ se calcula la magnitud del gradiente $m(x,\ y)$ y su orientación $\theta(x,\ y)$.

4. Se divide el rango de la orientación (0–180) en k contenedores. Luego se calcula el histograma de una celda (HC).

5. El histograma de un bloque (HB) puede ser obtenido integrando los histogramas de cada celda (HCs) dentro de ese bloque. Cada histograma de bloque se normaliza con la norma L2.

6. El histograma resultante HOG, puede ser obtenido integrando todos los histogramas de bloques normalizados.

3.2.3. HOF (Histogram of Optical Flow)

HOF se encarga de capturar la información de movimiento local. A diferencia de los anteriores, no puede ser calculado sobre una única imagen, ya que como se trata del flujo

se necesitan como mínimo dos. Para calcular este flujo óptico entre dos imágenes, hay que resolver la siguiente ecuación:

$$I_x u + I_y v + I_t = 0$$

- I_x, I_y e I_t son las derivadas de la luminosidad de las imágenes espacio-temporales.
- u es el flujo óptico horizontal.
- v es el flujo óptico vertical.

Para ésta tarea, hemos utilizado el método de **Horn-Schunck** [Horn and Schunck, 1981], el cual se basa en, asumiendo que el flujo óptico es regular sobre toda la imagen, calcular una estimación del campo velocidad $[u\, v]$ para cada pixel en la imagen. Para obtener u y v usando éste método los pasos son:

1. Se calculan I_x e I_y usando el kernel de convolución de *Sobel*, $[-1\ -2\ -1;\ 0\ 0\ 0;\ 1\ 2\ 1]$ y su forma traspuesta para cada pixel en la primera imagen.
2. Se calcula I_t entre las dos imágenes usando el kernel $[-1\ 1]$.
3. Se asume que la velocidad previa es 0 y se calcula la velocidad media por cada pixel usando $[0\ 1\ 0;\ 1\ 0\ 1;\ 0\ 1\ 0]$ como kernel de convolución.
4. Se itera para resolver u y v.

3.2.4. MBH (Motion Boundary Histogram)

Este descriptor fue propuesto para la detección de personas calculando por separado las derivadas para las componentes horizontal y vertical del flujo óptico. El descriptor codifica el movimiento relativo entre pixels. Dado que MBH representa el gradiente del flujo óptico, el movimiento constante y local de la cámara se evita mientras que la información sobre cambios en el campo del flujo se mantiene. MBH es más robusto al movimiento de la cámara que el flujo óptico y por lo tanto más discriminativo para el reconocimiento de acciones. Los pasos principales serían:

1. Se separa el flujo óptico $w = (u, v)$ en sus componentes horizontal y vertical.
2. Se calculan las derivadas espaciales para cada una de ellas.
3. La información sobre la orientación es cuantificada en histogramas.

4. Los histogramas obtenidos para cada componente son normalizados separadamente con su norma L_2.

Como el cálculo de éste descriptor se basa también en el flujo óptico, se ha utilizado el mismo método que para el descriptor HOF explicado anteriormente.

3.3. Requisitos (Funcionales y No funcionales)

El proyecto aborda la realización de un prototipo para probar caracterizadores visuales sobre vídeos en primera persona. Con el objetivo de integrar con facilidad los desarrollos previos de los trabajos de investigación analizados en el capítulo 2, se utilizará Matlab como plataforma de desarrollo.

Funcionales

RF1 Dado un dataset de vídeos, el sistema permitirá extraer un porcentaje determinado de frames de cada vídeo y almacenarlos.

RF2 El sistema permitirá probar los descriptores SURF, HOG, HOF y MBH.

RF3 El sistema permitirá probar diferentes datasets de distintas fuentes y tamaños.

RF4 El sistema permitirá reconocer objetos tales como cañones de nieve, vallas, balizas, personas y sombras en vídeos de snowboard con cámara subjetiva.

RF5 El sistema proporcionará un porcentaje de acierto junto con una matriz de confusión para poder comparar los resultados entre los distintos descriptores y datasets.

RF6 El sistema devolverá frames de los vídeos utilizados donde se resalten las zonas que ha tenido en cuenta cada descriptor.

RF7 El sistema permitirá añadir nuevos descriptores visuales de forma sencilla.

No Funcionales

RNF1 Los vídeos estarán en formato AVI o MP4.

RNF2 El prototipo se desarrolla íntegramente con Matlab (R2015a).

RNF3 El formato de los frames debe ser JPG.

Capítulo 4

Prototipo del sistema

Según lo presentado en la Figura 3.1 de la arquitectura del sistema, el prototipo está organizado de la siguiente manera.

4.1. Creación Datasets

En la primera parte se procede a crear los datasets (Figura 4.1). A partir de un conjunto considerable de vídeos, se clasifican y se recortan aquellos instantes en los cuales aparecen las clases que queremos identificar.

4.2. Extracción de Frames

Una vez obtenidos estos nuevos conjuntos de vídeos más detallados se procede a la extracción de sus frames (Figura 4.2). El porcentaje total que se extraigan dependerá de la longitud del vídeo y siempre serán frames aleatorios para aumentar la heterogeneidad del dataset.

4.3. Ejecución

Con nuestro conjunto de imágenes creado se selecciona el descriptor con el que se quiera trabajar, se crea el Bag Of Words a partir del dataset y se entrena un clasificador con dicho BoW que sea capaz de identificar las clases que utilizamos a partir de una imagen cualquiera (ver Figura 4.3 y Figura 4.4).

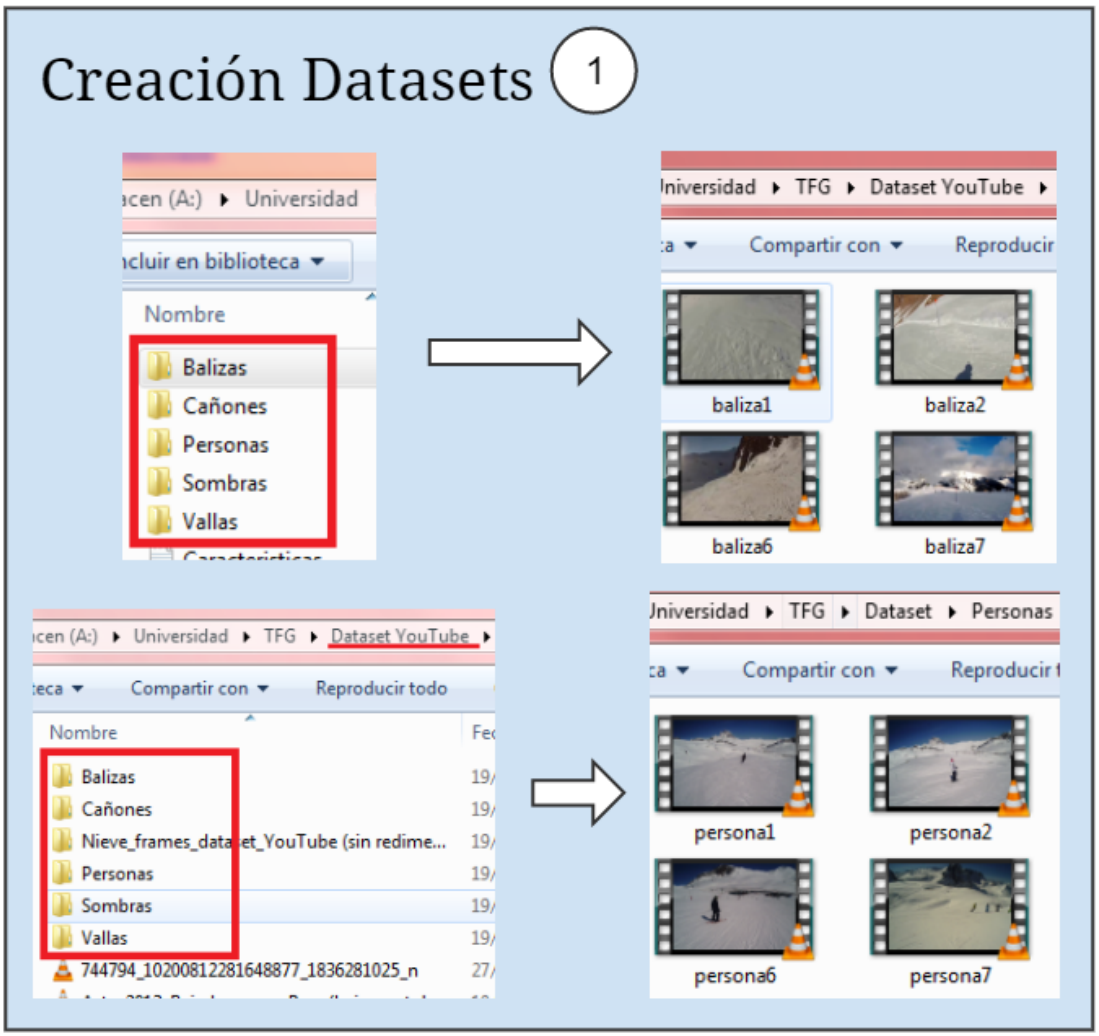


Figura 4.1: Primera parte: Creación

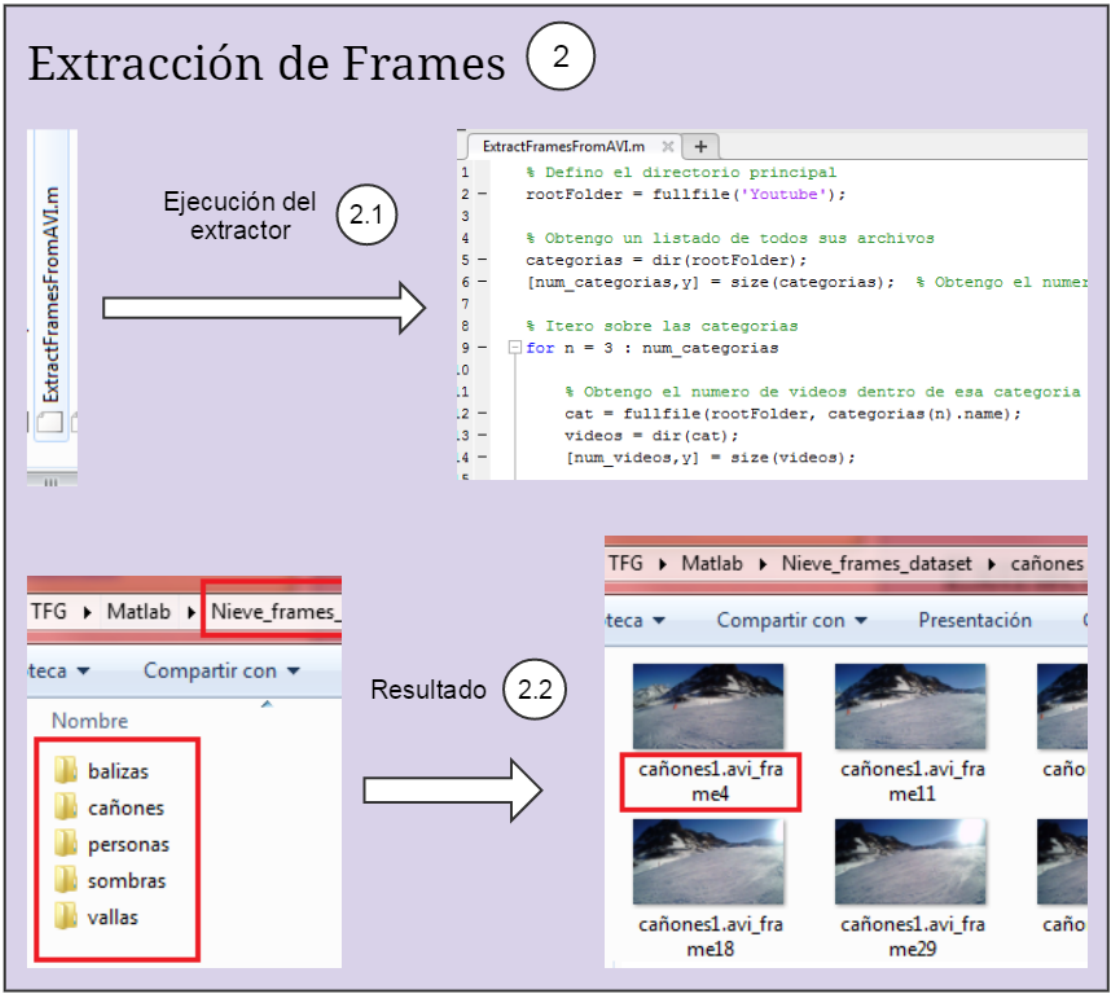


Figura 4.2: Segunda parte: Extracción

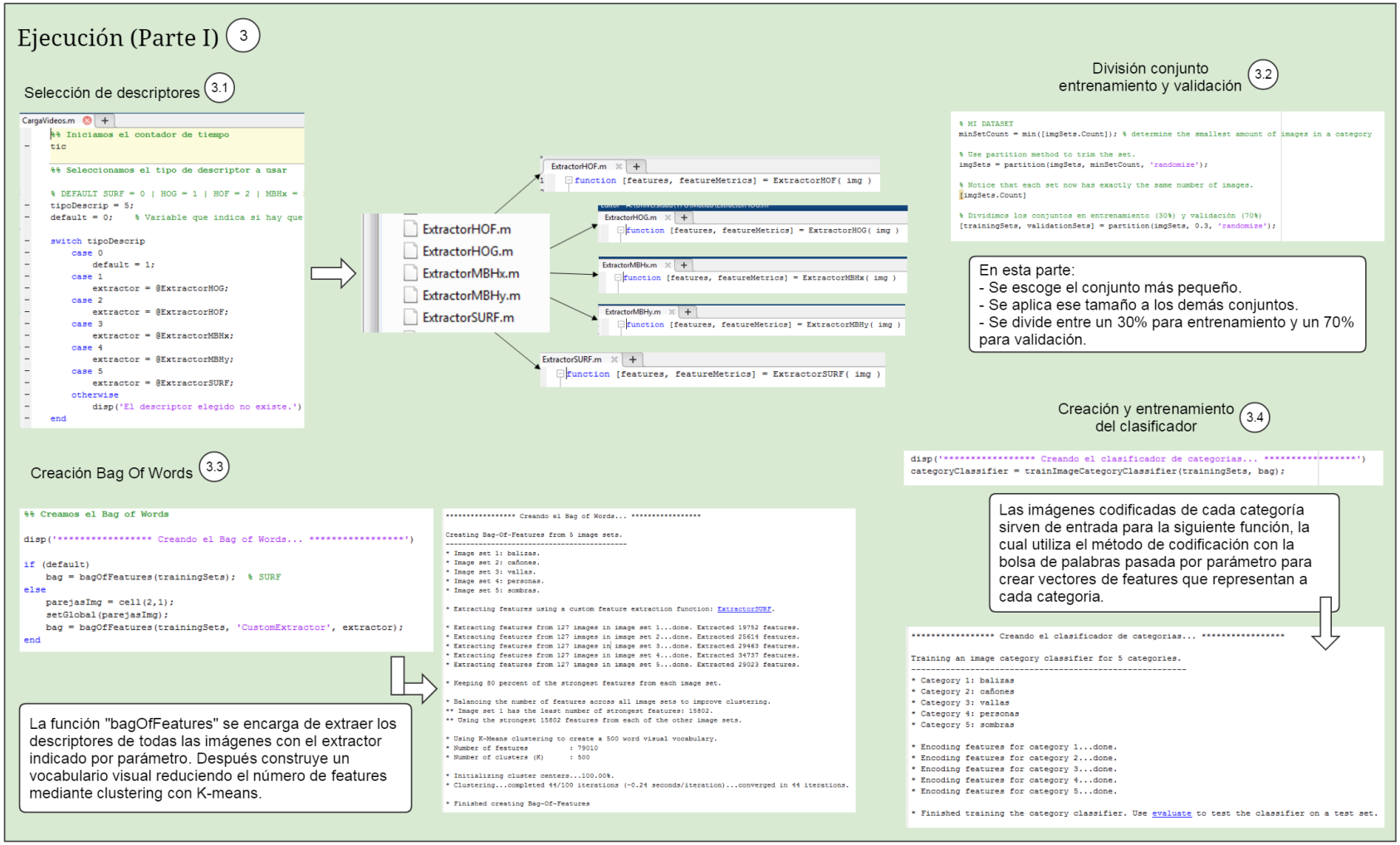


Figura 4.3: Tercera parte: Ejecución I

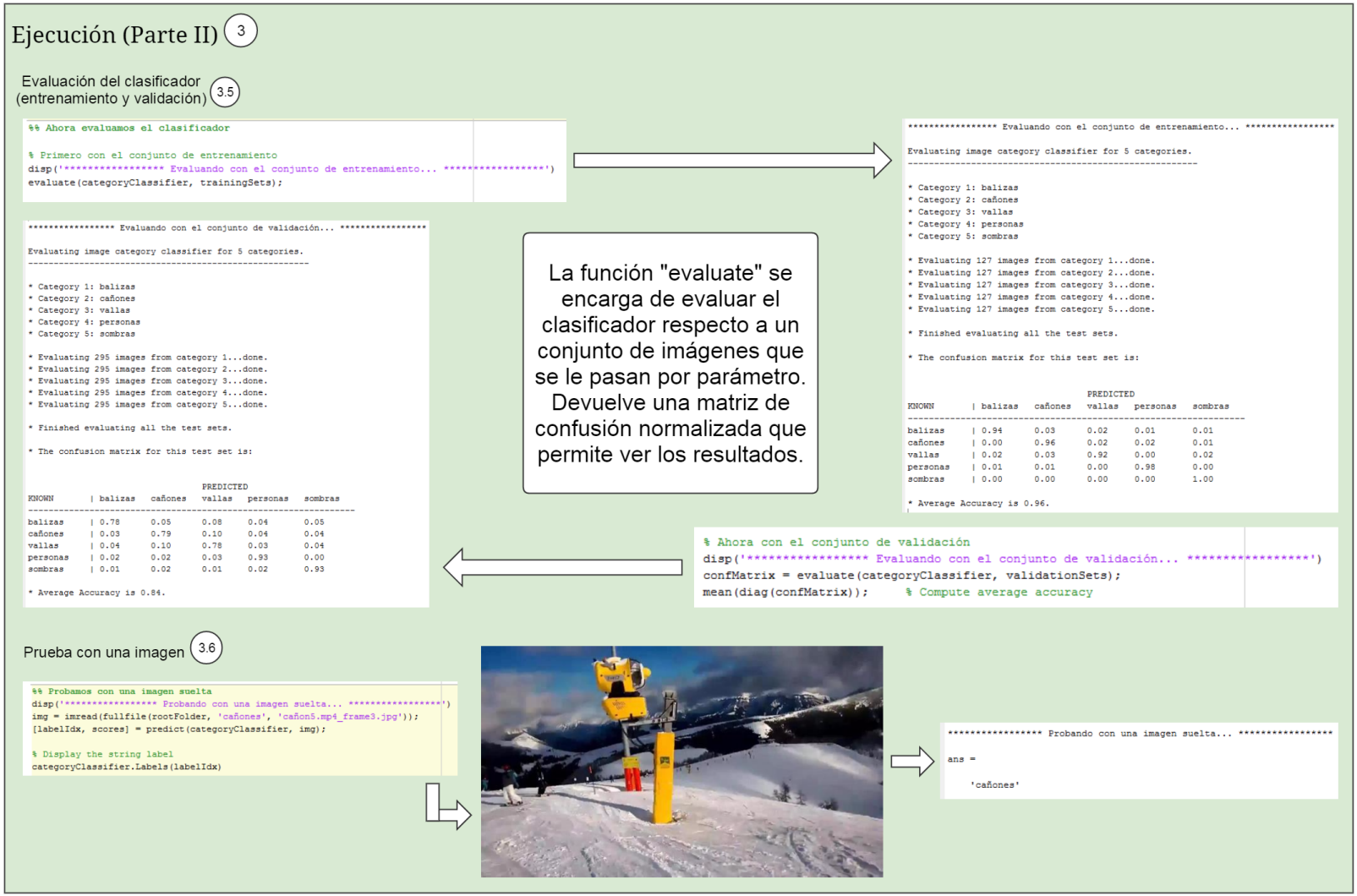


Figura 4.4: Tercera parte: Ejecución II

Capítulo 5

Resultados y Evaluación

En éste capítulo se va a proceder a realizar la descripción del experimento, detallando cada conjunto de datos utilizado y se mostrarán una serie de resultados los cuales serán evaluados.

5.1. Descripción del experimento

En este apartado se presenta el conjunto de pruebas realizadas con los descriptores HOG, HOF, MBH y SURF sobre un dataset propio (ver 5.1). Posteriormente se ha probado con un dataset de YouTube (ver 5.2) y finalmente con la mezcla de ambos (ver 5.3).

A continuación se muestran en detalle las características de los datasets utilizados.

Características / Datasets	**Cañones**	**Vallas**	**Balizas**	**Sombras**	**Personas**
Nº de vídeos	20	10	20	10	20
Duración total	1:21 min	1:11 min	0:44 min	1:13 min	1:31 min
Tamaño total	1.58 GB	1.38 GB	887 MB	1.42 GB	1.82 GB
Duración media[1]	3.65 seg	6.7 seg	1.7 seg	6.8 seg	4.3 seg
Porcentaje de frames[2]	25 %	30 %	44 %	28 %	22 %
Nº de frames	623	648	608	621	628

Cuadro 5.1: Tabla resumen del Dataset Inicial

En la tabla 5.1 se detalla el conjunto de datos recopilados de los vídeos grabados por mí. Este dataset es sin duda el más completo de todos ya que, como se puede observar, cada clase tiene un tamaño considerable. Por ello, se ha extraído un porcentaje pequeño de frames, pero suficiente para realizar las pruebas.

[1]La duración media representa la longitud promedia de todos los vídeos de este dataset.
[2]El porcentaje de frames es la cantidad de imágenes extraídas de los vídeos.

Características / Datasets	Cañones	Vallas	Balizas	Sombras	Personas
Nº de vídeos	9	8	9	10	7
Duración total	0:21 min	0:27 min	0:21 min	0:45 min	0:33 min
Tamaño total	54.8 MB	11 MB	5.88 MB	2705 MB	11.9 MB
Duración media[3]	2 seg	3 seg	2 seg	4.2 seg	4.4 seg
Porcentaje de frames[4]	90 %	90 %	90 %	90 %	90 %
Nº de frames	436	700	521	1230	863

Cuadro 5.2: Tabla resumen del Dataset de YouTube

En la tabla 5.2 se detalla el conjunto de datos recopilados de vídeos de la plataforma YouTube. A diferencia del dataset inicial, el dataset de YouTube cuenta con menos vídeos ya que fue más complicado recopilar aquellos con similares características de grabación. Por ello se ha extraído un porcentaje bastante más elevado.

Características / Datasets	Cañones	Vallas	Balizas	Sombras	Personas
Nº frames total	429	449	424	453	422
Nº frames Dataset Inicial	231	117	214	222	212
Nº frames Dataset YouTube	198	272	210	231	210

Cuadro 5.3: Tabla resumen del Dataset mezcla de los anteriores

En la tabla 5.3 se muestra el conjunto de datos que conforman una mezcla heterogénea de los dos anteriores. Dicha mezcla ha sido realizada a partir de los frames extraidos, no de sus vídeos.

Dado que ya se contaba con 2 datasets completos y con información muy diversa, se decidió crear uno nuevo que contuviera una mezcla de ambos en la misma proporción. Para de ésta manera, poder validar los dos anteriores y realizar un nuevo experimento con un mejor entrenamiento previo.

5.2. Resultados

A continuación se muestran una serie de gráficos generales que describen los resultados de la experimentación realizada y los tiempos obtenidos en ella. Todo el experimento ha sido realizado con un ordenador de sobremesa con las siguientes características:

- Sistema operativo Windows 7.
- Procesador Intel Core i5-3570 a 3.4GHz.
- Memoria RAM de 8 GB.

[3]La duración media representa la longitud promedia de todos los vídeos de este dataset.
[4]El porcentaje de frames es la cantidad de imágenes extraídas de los vídeos.

- Tarjeta gráfica AMD Radeon R9 380X.

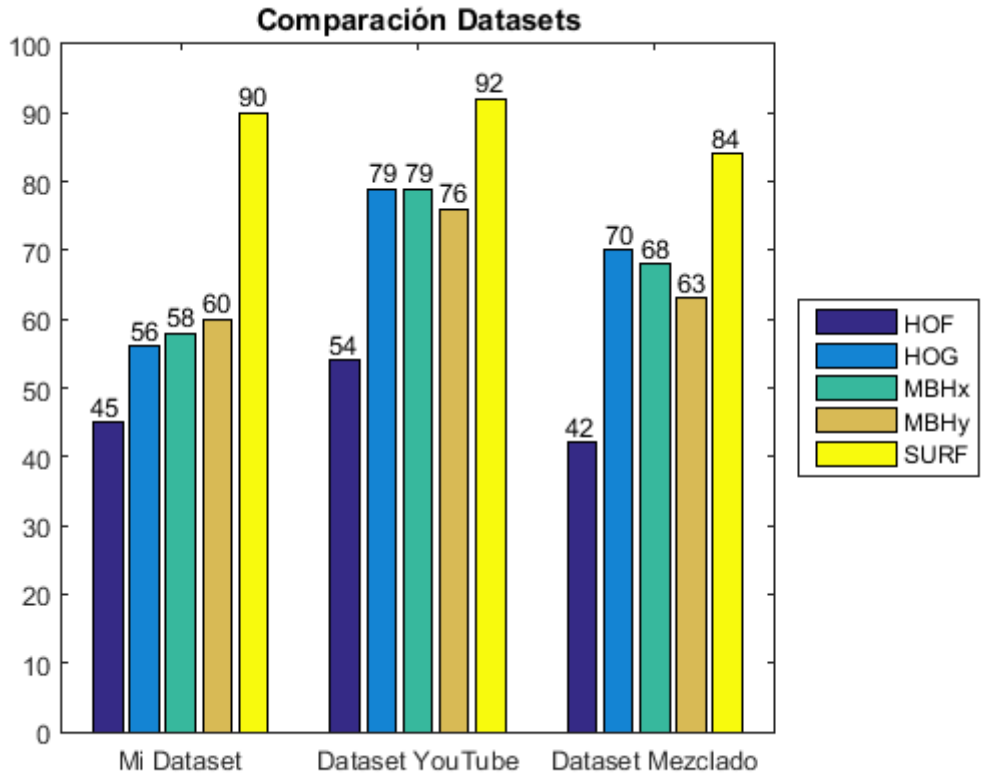


Figura 5.1: Comparación gráfica del rendimiento de cada descriptor en cada dataset.

La Figura 5.1 se ha construido promediando los resultados que ha tenido cada descriptor, sobre cada una de las clases, en la parte de validación. Como se puede observar, el descriptor que mejor ha funcionado en todos los datasets ha sido SURF, logrando incluso un 92 % de éxito en algunos casos. Mientras que HOF apenas ha logrado alcanzar el 50 %.

Para los gráficos 5.2, 5.3 y 5.4, se ha extraído el resultado en la parte de validación, de cada descriptor sobre cada clase.

En el caso concreto de la figura 5.2 del dataset inicial, se sigue viendo que SURF se diferencia notablemente de los demás, pero la gráfica permite ver que hay determinadas clases que se reconocen mejor que otras, como las balizas o las sombras.

En el gráfico 5.3 se obtienen unos resultados considerablemente mejores que para el caso anterior. Aunque se mantiene el mismo patrón que antes, dado que las sombras y balizas siguen siendo las clases que mejor reconocen todos los descriptores.

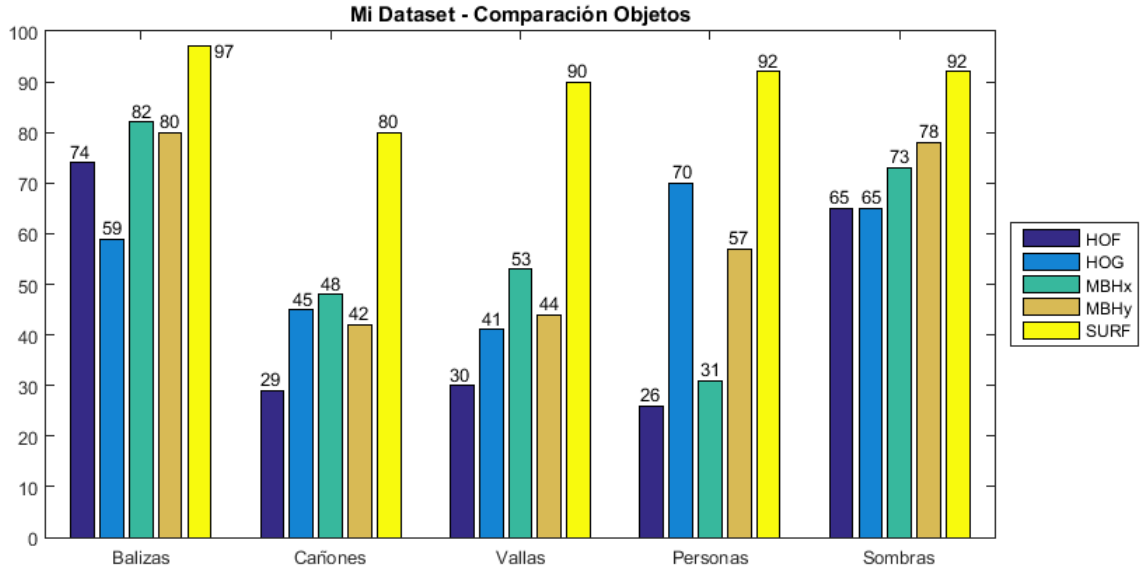


Figura 5.2: Comparación gráfica del rendimiento de cada descriptor en las clases del dataset inicial.

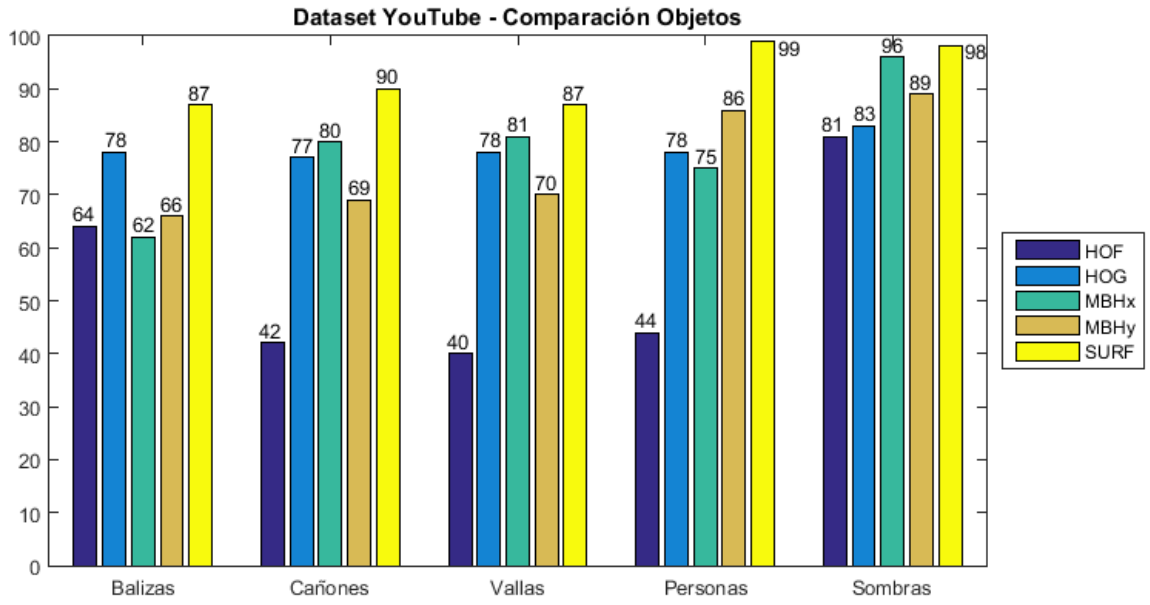


Figura 5.3: Comparación gráfica del rendimiento de cada descriptor en las clases del dataset de YouTube.

En la figura 5.4, al tratarse de un dataset compuesto por los dos anteriores, viene a ser una media de ambos. Dado que para cada clase, al menos 3 descriptores han dado valores intermedios a los que habían proporcionado para los anteriores datasets.

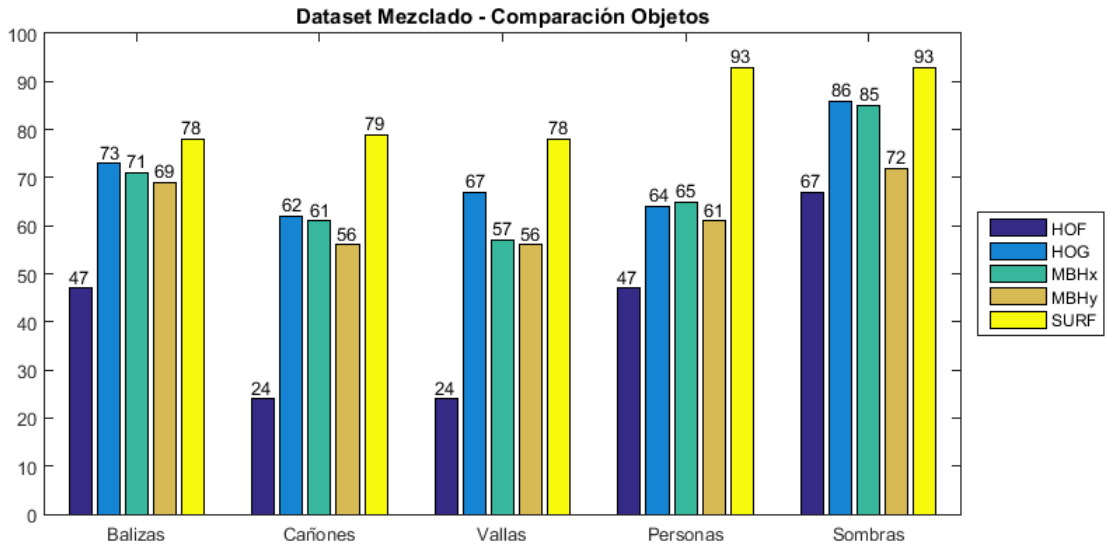


Figura 5.4: Comparación gráfica del rendimiento de cada descriptor en las clases del dataset mezclado.

Imagen	HOG	HOF	MBHx	MBHy	SURF
	0.030473 segundos	0.013340 segundos	0.014835 segundos	0.014919 segundos	0.019496 segundos

Cuadro 5.4: Tabla de tiempos para una imagen

Descriptores / Datasets	Dataset Inicial (3040 imágenes)	Dataset YouTube (2180 imágenes)	Dataset Mezclado (2110 imágenes)
HOG	4'468430"	3'092063"	3'087647"
HOF	1'711677"	1'339426"	1'203989"
MBHx	1'710903"	1'339426"	1'150302"
MBHy	1'681631"	1'222322"	1'156047"
SURF	2'596146"	1'749060"	1'865899"

Cuadro 5.5: Tabla de tiempos general

Por último, en las tablas de tiempos 5.5 y 5.4, se observa que HOG es el descriptor visual que más tarda (celdas azules) aunque también es el segundo que mejor clasifica. Por otro lado los MBH's y HOF son los más rápidos aunque sus resultados no sean tan buenos (celdas verdes).

5.3. Evaluación

Como conclusión de estos resultados se puede extraer que el dataset de YouTube tiene una tasa mayor de aciertos para todos los descriptores. Pienso que esto es debido a que la experiencia previa en el montaje del dataset inicial con mis imágenes, ha servido para establecer unos patrones de trabajo que han mejorado la calidad en la posterior construcción del dataset de vídeos de Youtube, obteniendo un mayor detalle.

En cuanto a los descriptores, SURF ha resultado el mejor de todos ellos para las pruebas realizadas. Por otro lado HOF sería el peor, aunque en determinados casos como el reconocimiento de balizas o sombras da resultados aceptables. Para el resto no se obtienen resultados concluyentes ya que varían en función de la clase o el dataset probado.

En cuanto a los tiempos obtenidos, la complejidad temporal del algoritmo SURF de Matlab es notable frente al resto de descriptores probados. Resulta patente que la combinación de transformaciones globales sobre la imagen, el cálculo de la Hessiana, la creación de múltiples descriptores sobre regiones pequeñas de la imagen y el emparejamiento de puntos característicos, requieren mucho procesamiento global y local. No obstante, las mejoras en cuanto a la calidad de los resultados frente al resto de los descriptores justifican su utilización.

Para ampliar la información acerca de los resultados se puede consultar el Apéndice A.

Capítulo 6

Gestión del proyecto y Conclusiones

A continuación se abordará todo lo relacionado con la gestión de este estudio y se finalizará con unas conclusiones generales.

6.1. Gestión del proyecto

En esta sección se van a detallar las diferentes tareas que se han realizado y el tiempo que se ha dedicado a cada una de ellas.

El desarrollo del proyecto ha estado sujeto a circunstancias que me gustaría hacer constar. Por un lado, en cuanto a la parte técnica, se planteó desde un principio realizar una experimentación más elaborada, utilizando técnicas de combinación de descriptores para mejorar los resultados individuales de cada uno de ellos. Esta parte no pudo llevarse a cabo ya que no resultó posible acceder a modificar las funciones básicas de bajo nivel de Matlab, las cuales estaban implicadas en el procesamiento de matrices que combinaban los distintos descriptores visuales.

Adicionalmente, se detectaron inconsistencias entre la misma versión en distintas plataformas (Windows y Mac), produciéndose resultados diferentes, que no permitían usar directamente el código suministrado por Matlab con las garantías necesarias para decidir sobre las posibles mejoras derivadas de la concatenación de descriptores.

Esto consumió varias semanas en el desarrollo del proyecto que han quedado correspondientemente reflejadas en el diagrama de GANTT (Figura 6.1) y detalladas en el Apéndice C.

Por otro lado, mis circunstancias particulares han condicionado el desarrollo del proyecto ya que he simultaneado su desarrollo con prácticas de empresa (3 + 3 meses).

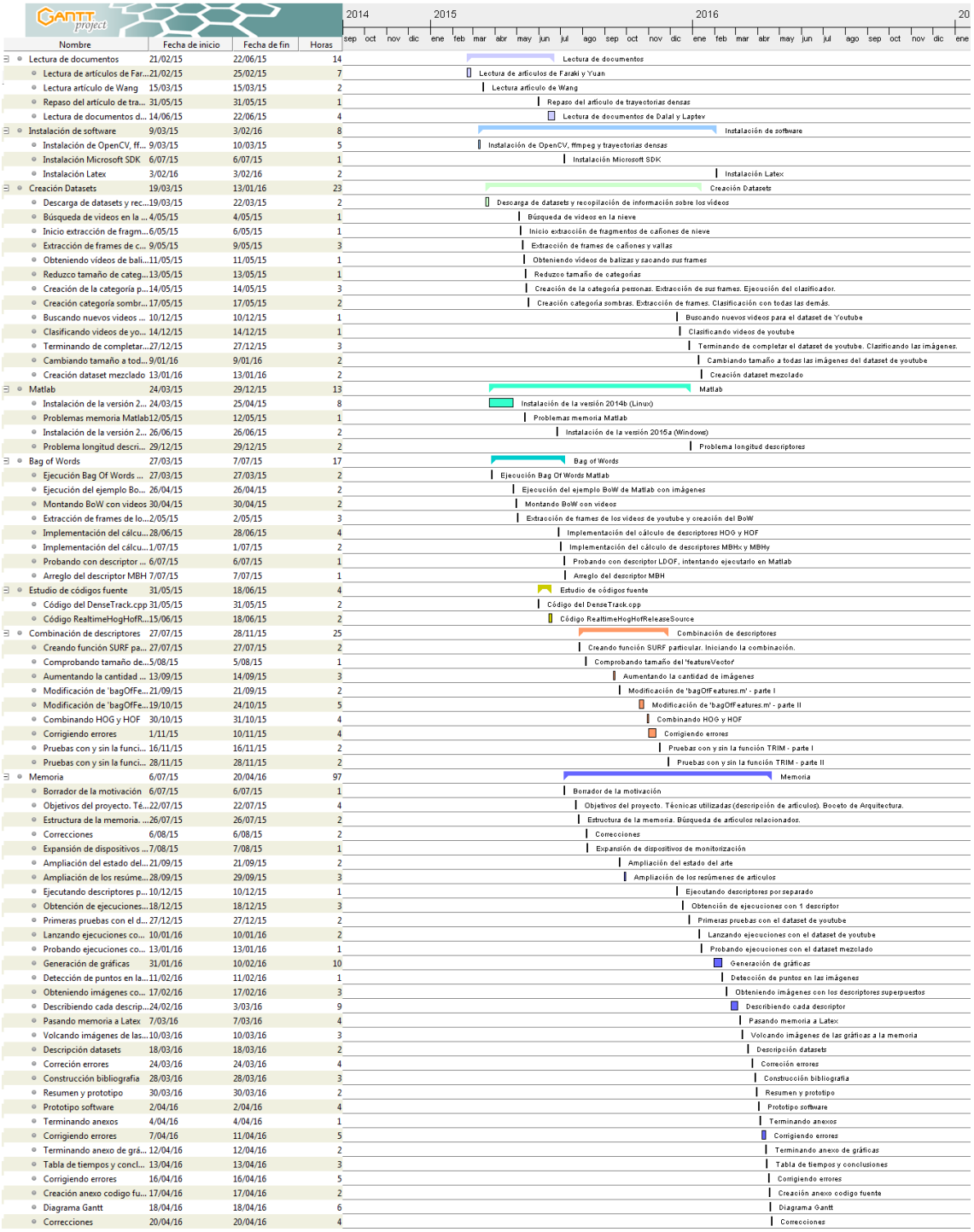


Figura 6.1: Diagrama de Gantt

6.2. Conclusiones

Este estudio ha servido para ampliar el ámbito de uso de las técnicas de aprendizaje hacia un dominio de aplicaciones relacionadas con la monitorización de la actividad deportiva.

A la vista está que los resultados obtenidos son prometedores y lo observado podría utilizarse como estudio previo para desarrollar dispositivos portátiles que tuvieran una aplicación directa en este deporte.

En lo que se refiere a la experimentación, se puede concluir que el descriptor visual SURF es el que mejor ha funcionado en todos los casos, para cualquier dataset y categoría de objeto a reconocer.

Los resultados también han demostrado que dependen mucho de los conjuntos de datos utilizados. Para datasets como el de YouTube donde las imágenes están mejor clasificadas e identificadas como las diferentes categorías de clasificación y con mayor calidad, los resultados son considerablemente mejores.

De modo que un futuro desarrollo podría estar centrado en dos aspectos: crear un dataset mejorado que los utilizados en este estudio y optimizar el descriptor SURF para lograr reducciones significativas en el tiempo de cálculo del mismo. Éstos podrían ser los pasos a seguir para poder llegar a integrar el prototipo en un dispositivo.

Apéndice A

Resultados gráficos detallados

A.1. Dataset Inicial

Los siguientes resultados muestran el nivel de acierto para cada descriptor usado con este dataset. Cada clase utilizada está evaluada con un conjunto de entrenamiento y otro de validación.

Utilizando este conjunto de datos podemos ver que HOF no es uno de los mejores descriptores pero vemos que funciona correctamente identificando balizas y sombras (ver figura A.1).

HOG proporciona unos resultados más homogéneos aunque tampoco muy esperanzadores. Aun así consigue superar el 50 % de acierto en validación para balizas, personas y sombras (ver figura A.2).

MBHx logra una clasificación muy buena para balizas y sombras superando a HOF. Los resultados se repiten para MBHy con ligeras variaciones en la detección de personas y vallas.

SURF sin embargo, supera a todos los anteriores siendo su mínimo valor el 80 % de acierto en el conjunto de validación sobre cañones, que aun así sigue siendo el máximo alcanzado respecto a los demás (ver figura A.5).

Para todos los datasets se ha obtenido una última gráfica que representa el nivel de acierto promedio de cada descriptor sobre todas las clases utilizadas. Para este caso en concreto, sin duda alguna SURF da los mejores resultados, con una considerable diferencia respecto al segundo mejor (ver figura A.6).

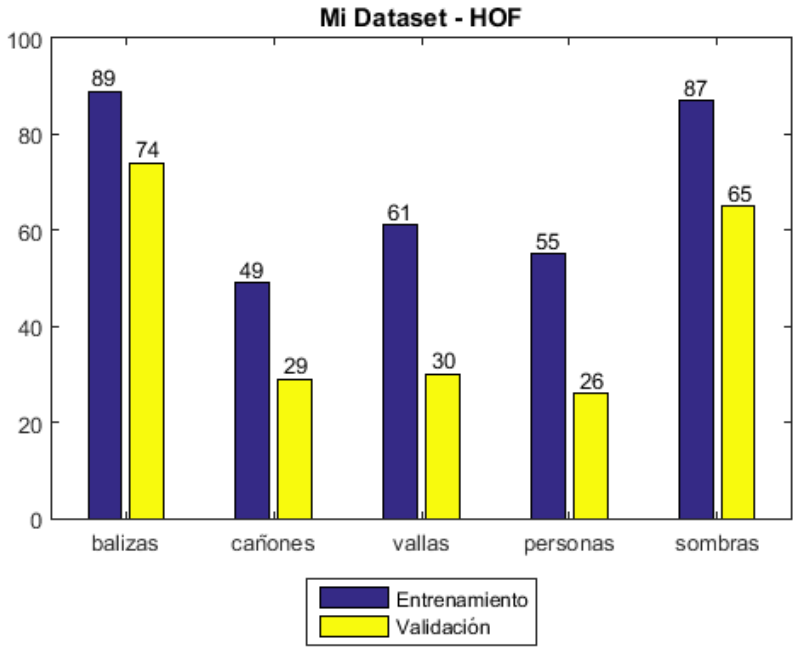


Figura A.1: Resultados del descriptor HOF.

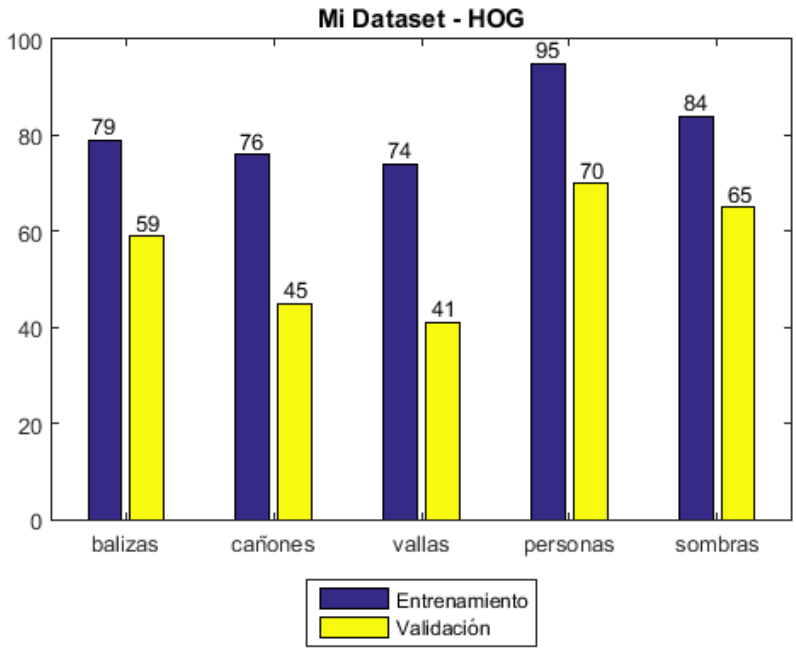


Figura A.2: Resultados del descriptor HOG.

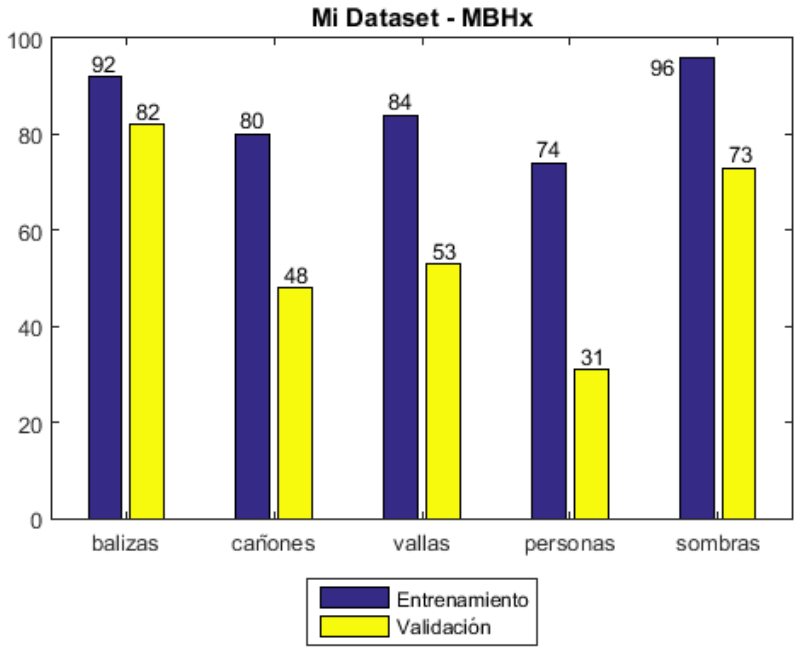


Figura A.3: Resultados del descriptor MBHx.

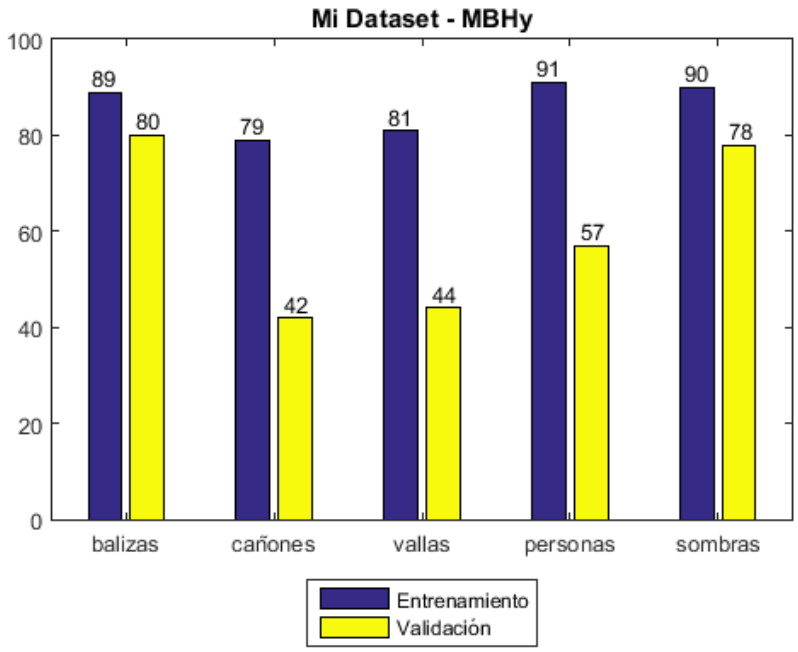


Figura A.4: Resultados del descriptor MBHy.

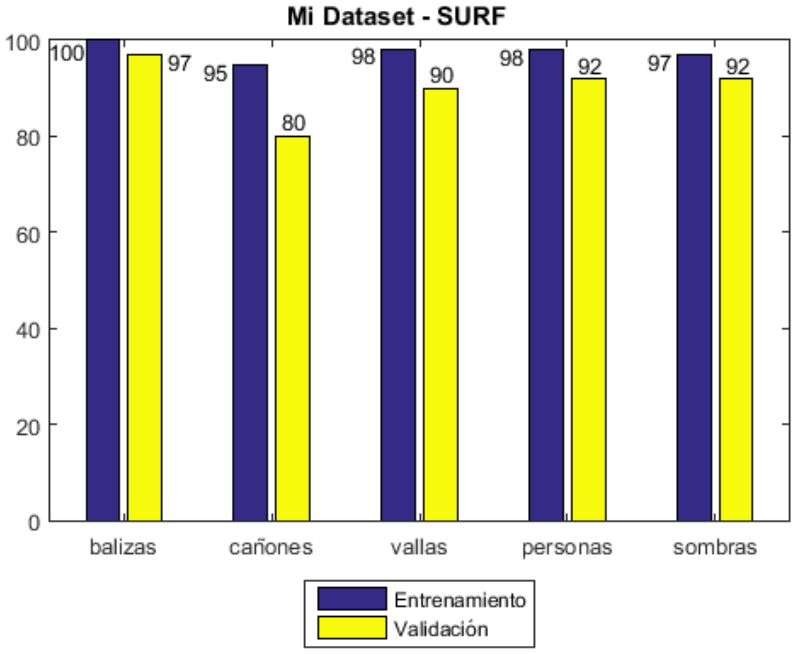


Figura A.5: Resultados del descriptor SURF.

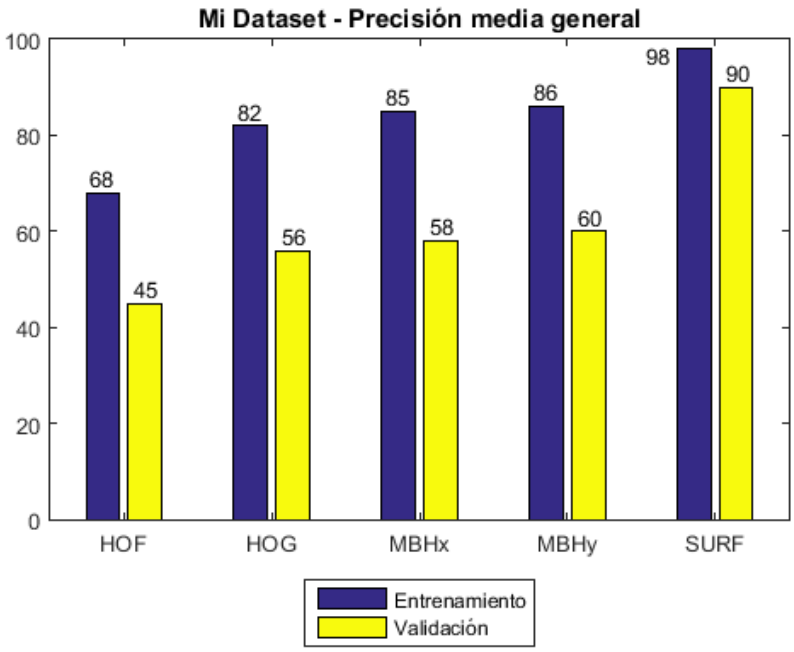


Figura A.6: Resultados de la precisión media general.

A.2. Dataset YouTube

Los siguientes resultados muestran el nivel de acierto para cada descriptor usado con este dataset. Cada clase utilizada está evaluada con un conjunto de entrenamiento y otro de validación.

Para el conjunto de imágenes de YouTube, HOF mejora, en comparación al dataset inicial, en el reconocimiento de todas las clases salvo en las balizas que baja un 10 % (ver figura A.7).

En el caso de HOG también se produce una mejora considerable para todas las clases llegando a alcanzar un 83 % de precisión para el conjunto de validación en algunos casos (ver figura A.8).

MBHx sufre una caída en el reconocimiento de balizas, sin embargo logra unos excelentes resultados para las sombras, aumentando también en las demás clases (ver figura A.9).

En MBHy se reproduce el mismo comportamiento que en el anterior con ligeras variaciones (ver figura A.10).

Por último, el descriptor SURF sigue prácticamente invariante respecto a sus resultados, proporcionando una alta precisión para todos los casos, rondando el 90 % para el conjunto de validación (ver figura A.11).

En la gráfica de la precisión media, se puede observar que todos los descriptores han mejorado respecto al conjunto de datos anterior. SURF es mejor respecto a los demás pero la diferencia se acorta ya que HOG y MBHx están rozando el 80 % en validación (ver figura A.12).

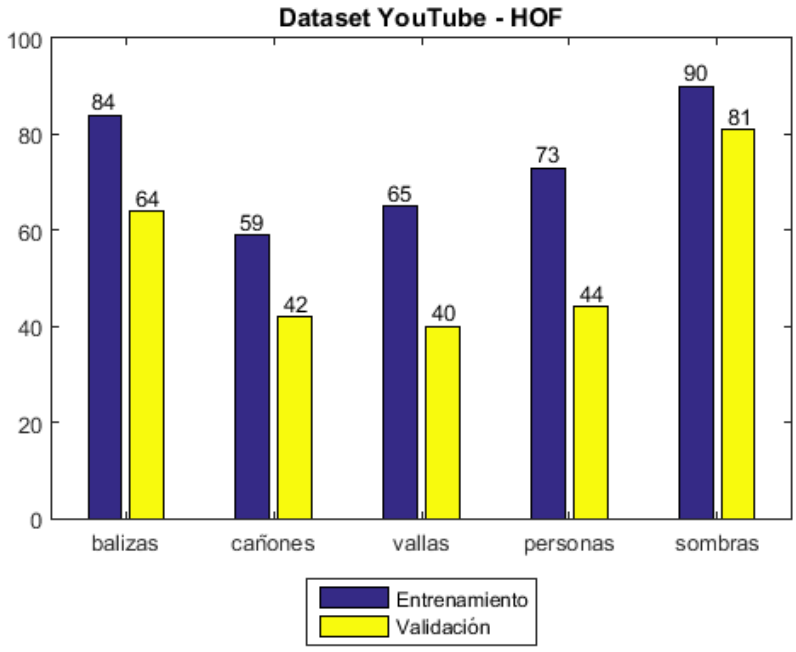


Figura A.7: Resultados del descriptor HOF.

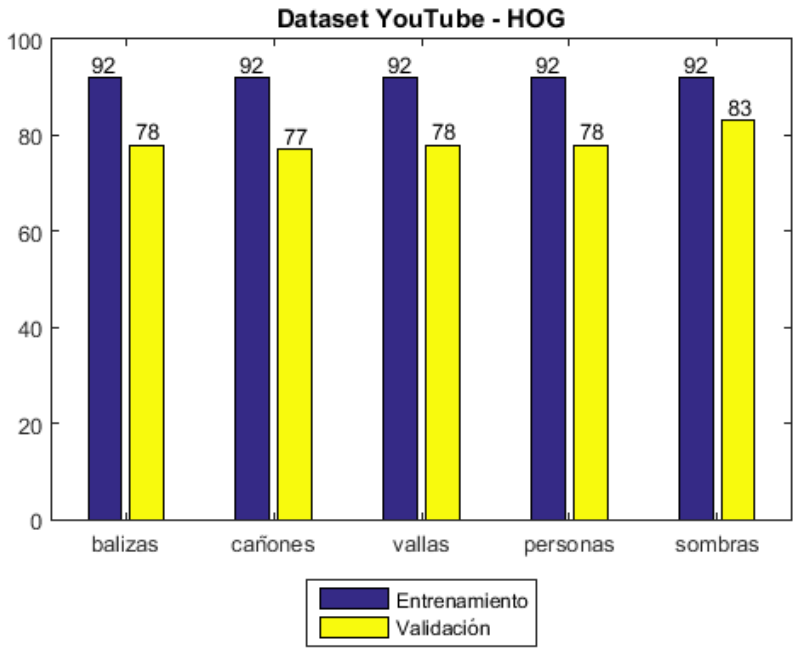


Figura A.8: Resultados del descriptor HOG.

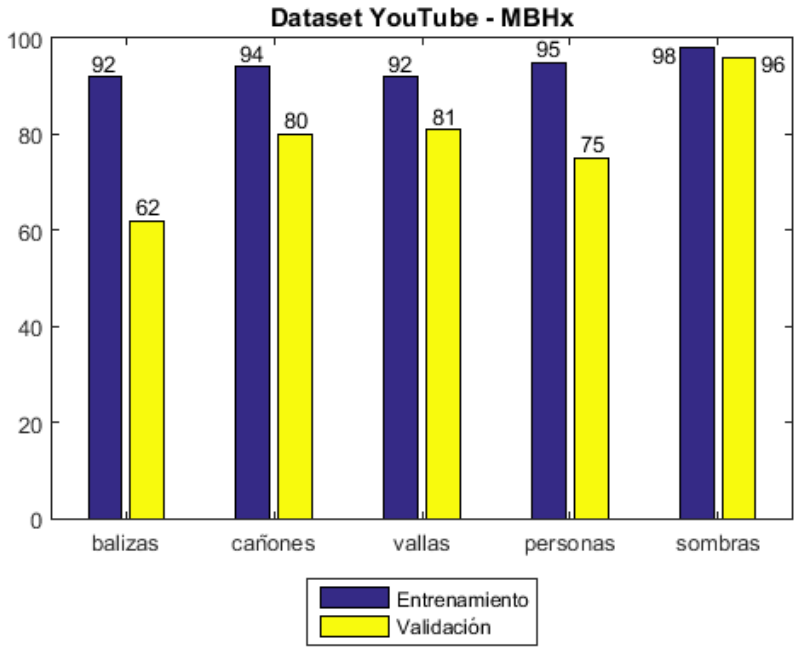


Figura A.9: Resultados del descriptor MBHx.

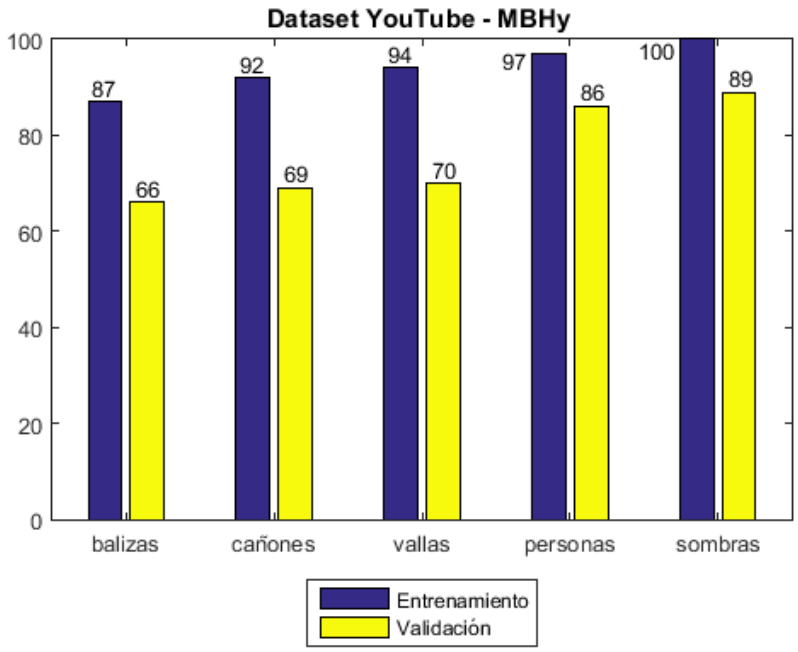


Figura A.10: Resultados del descriptor MBHy.

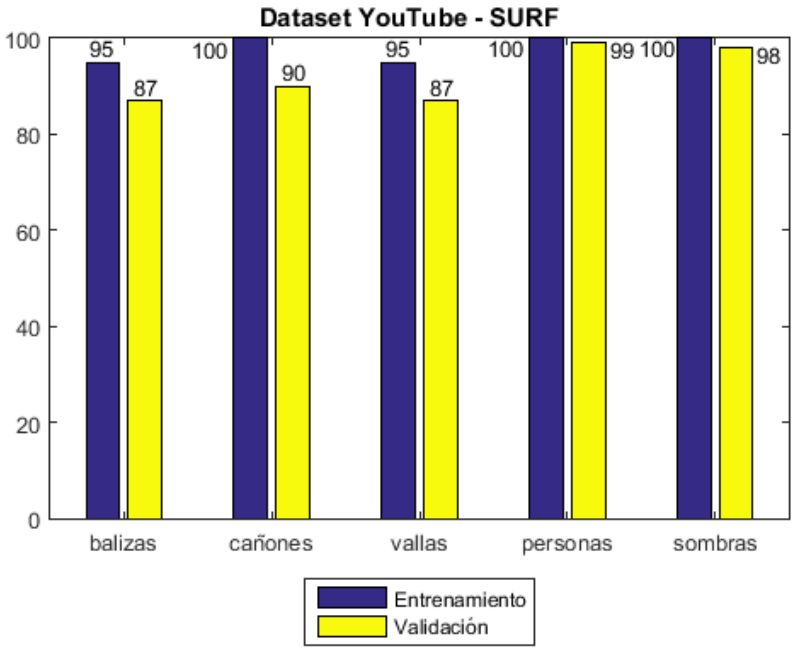


Figura A.11: Resultados del descriptor SURF.

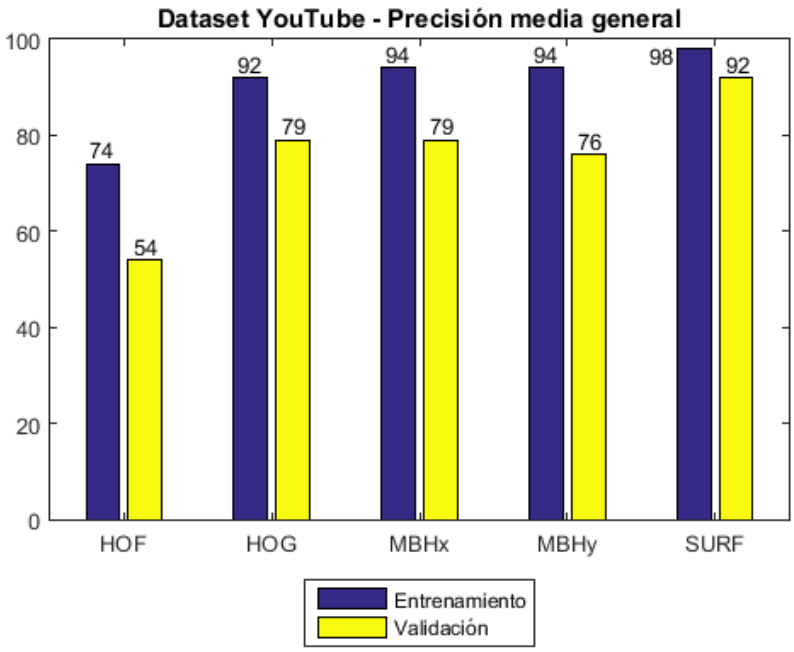


Figura A.12: Resultados de la precisión media general.

A.3. Dataset Mezclado

Los siguientes resultados muestran el nivel de acierto para cada descriptor usado con este dataset. Cada clase utilizada está evaluada con un conjunto de entrenamiento y otro de validación.

Respecto a HOF, se puede ver que los valores de precisión en sus resultados, están entre los dos casos anteriores. Por ejemplo, empeora sus resultados reconociendo cañones y vallas, pero mejora con las personas y tiene un valor intermedio en las sombras (ver figura A.13).

En el caso de HOG se constata más el hecho de que son resultados intermedios ya que todos los valores devueltos están entre los valores del dataset inicial y el de YouTube, habiendo una ligera mejora en el reconocimiento de sombras. Siempre respecto al conjunto de datos de validación (ver figura A.14).

Para MBHx y MBHy sigue ocurriendo lo mismo, todos y cada uno de sus valores son un resultado intermedio, salvo para el reconocimiento de sombras en MBHy que se da el valor más bajo entre los tres conjuntos (ver figura A.15 y figura A.16).

SURF sin embargo, ve empeorados sus resultados en las clases balizas, cañones y vallas bajando por debajo del 80 % y se mantiene en valores intermedios para las personas y las sombras (ver figura A.17).

Finalmente, la precisión media general muestra una caída para HOF y SURF respecto a los dos datasets anteriores, mientras que HOG, MBHx y MBHy se han mantenido en valores intermedios (ver figura A.18).

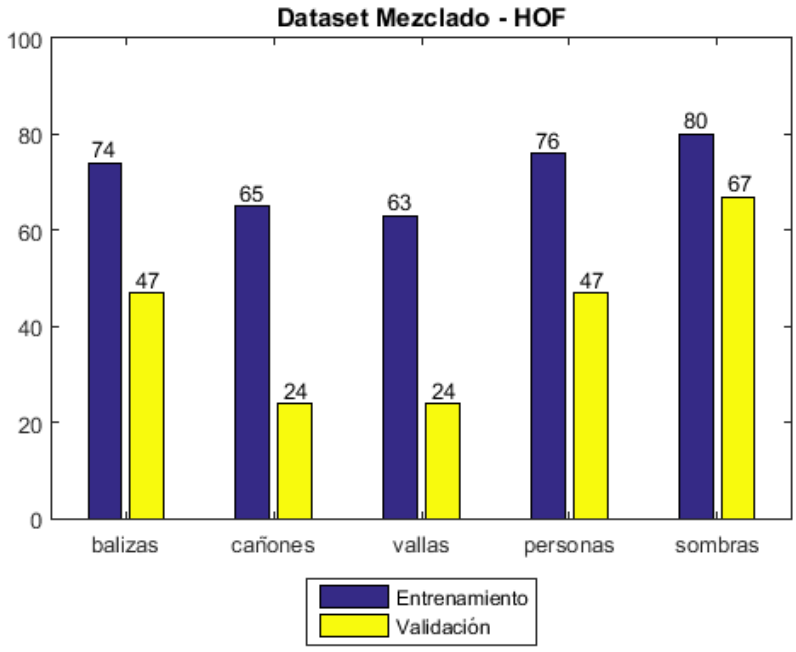


Figura A.13: Resultados del descriptor HOF.

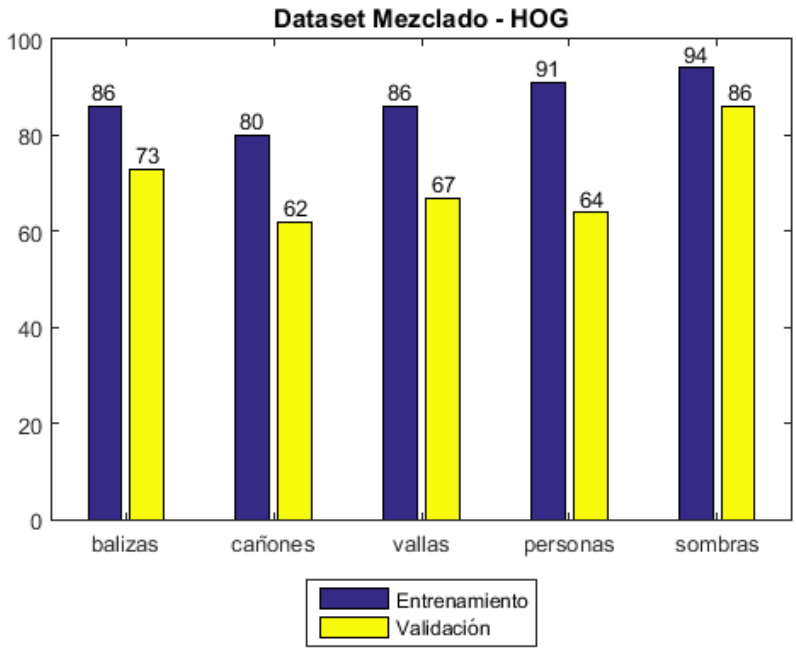


Figura A.14: Resultados del descriptor HOG.

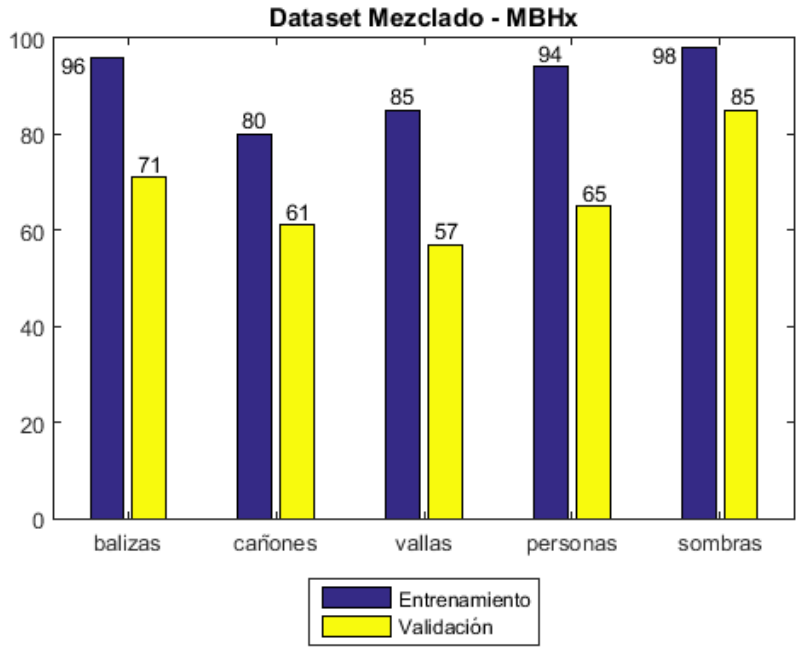


Figura A.15: Resultados del descriptor MBHx.

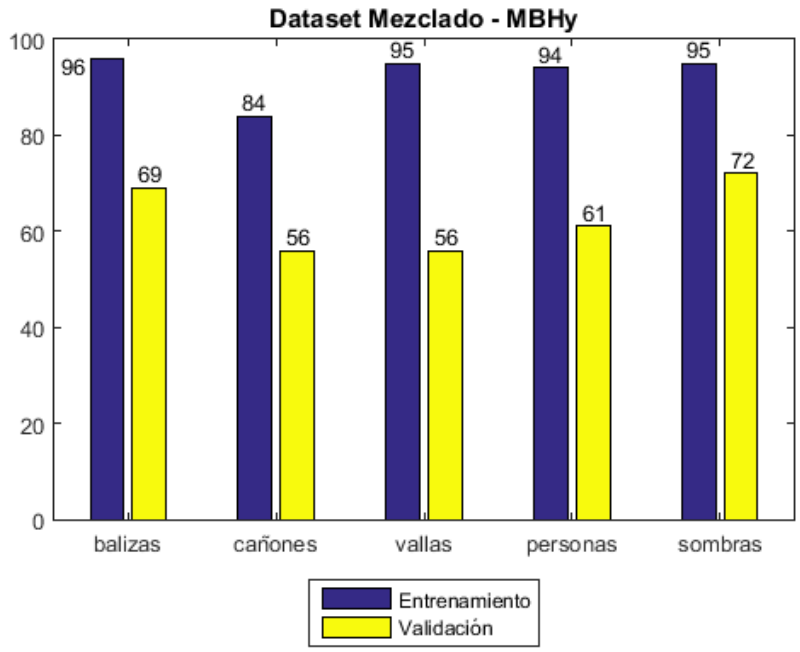


Figura A.16: Resultados del descriptor MBHy.

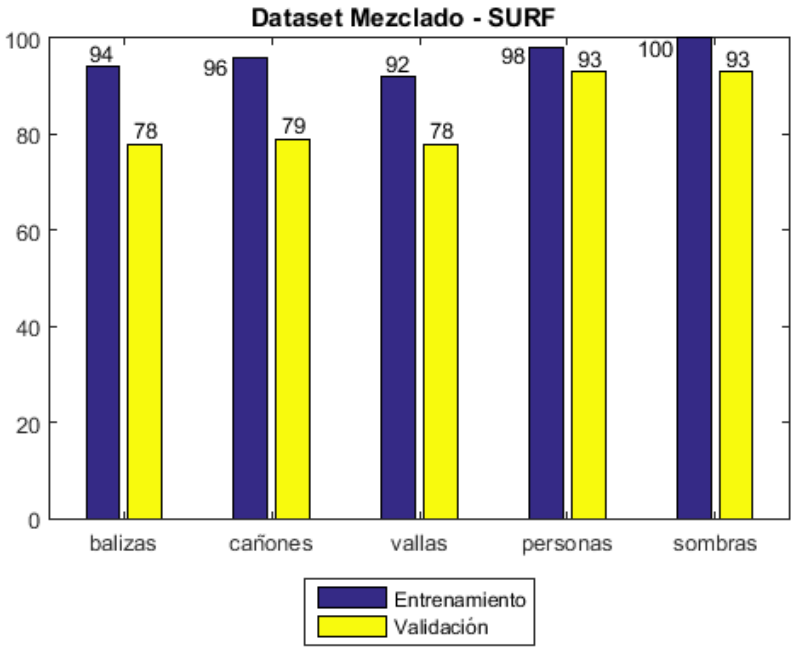


Figura A.17: Resultados del descriptor SURF.

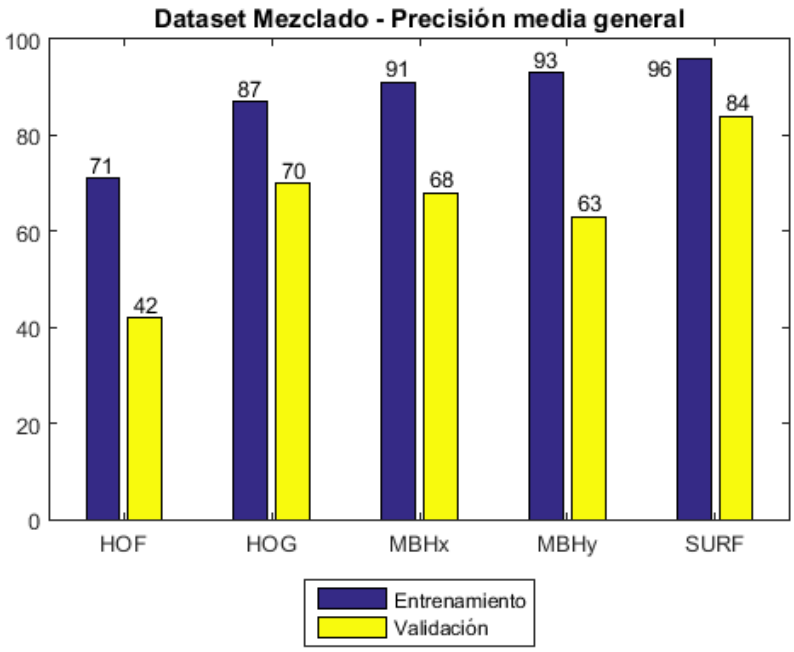


Figura A.18: Resultados de la precisión media general.

Apéndice B

Código fuente

B.1. CargaVideos.m

Se trata del script principal donde se lee el conjunto de imágenes, se elige el descriptor y se crea un clasificador entrenado y validado.

```
1  %% Iniciamos el contador de tiempo
2  tic
3
4  %% Seleccionamos el tipo de descriptor a usar
5
6  % DEFAULT SURF = 0 | HOG = 1 | HOF = 2 | MBHx = 3 | MBHy = 4 | SURF = 5
7  tipoDescrip = 5;
8  default = 0;   % Se ejecuta el 'bagOfFeatures' con un descriptor propio o no
9
10 switch tipoDescrip
11     case 0
12         default = 1;
13     case 1
14         extractor = @ExtractorHOG;
15     case 2
16         extractor = @ExtractorHOF;
17     case 3
18         extractor = @ExtractorMBHx;
19     case 4
20         extractor = @ExtractorMBHy;
21     case 5
22         extractor = @ExtractorSURF;
23     otherwise
24         disp('El descriptor elegido no existe.')
```

```
25 end
26
27 %% Load Video Sets
28
29     % Cargo el dataset
30     rootFolder = fullfile('Nieve_frames_dataset');
31
32     imgSets = [ imageSet(fullfile(rootFolder, 'balizas')), ...
33                 imageSet(fullfile(rootFolder, 'canones')), ...
34                 imageSet(fullfile(rootFolder, 'vallas')), ...
35                 imageSet(fullfile(rootFolder, 'personas')), ...
36                 imageSet(fullfile(rootFolder, 'sombras'))];
37
38     { imgSets.Description } % display all labels on one line
39     [imgSets.Count]         % show the corresponding count of images
40
41 %% Prepare Training and Validation Image Sets
42
43     % Determine the smallest amount of images in a category
44     minSetCount = min([imgSets.Count]);
45
46     % Use partition method to trim the set.
47     imgSets = partition(imgSets, minSetCount, 'randomize');
48
49     % Notice that each set now has exactly the same number of images.
50     [imgSets.Count]
51
52     % Dividimos los conjuntos en entrenamiento (30 %) y validacion (70 %)
53     [trainingSets, validationSets] = partition(imgSets, 0.3, 'randomize');
54
55 %% Creamos el Bag of Words
56
57 % La funcion 'bagOfFeatures' extrae los descriptores de todas las imagenes
58 % con SURF.
59 % Despues construye un vocabulario visual reduciendo el numero de features
60 % mediante clustering con K-means.
61 disp('***************** Creando el Bag of Words... ****************')
62
63 if (default)
64     bag = bagOfFeatures(trainingSets);   % SURF
65 else
66     parejasImg = cell(2,1);
67     setGlobal(parejasImg);
68     bag = bagOfFeatures(trainingSets, 'CustomExtractor', extractor);
69 end
70
71 % Esta funcion permite utilizar el metodo 'encode' para contar las
```

```
72  % ocurrencias de una visual word dentro de una imagen. Produce un
73  % histograma que se convierte en una nueva y reducida representacion de la
74  % imagen.
75   img = read(imgSets(1), 1); % Escogemos una imagen cualquiera
76   featureVector = encode(bag, img);
77
78  % Mostramos el histograma que sirve para codificar la imagen en un vector
79  % de features.
80   figure
81   bar(featureVector)
82   title('Visual word occurrences')
83   xlabel('Visual word index')
84   ylabel('Frequency of occurrence')
85
86  % Las imagenes codificadas de cada categoria sirven de entrada para la
87  % siguiente funcion, la cual utiliza el metodo de codificacion con la
88  % bolsa de palabras pasada por parametro para crear vectores de features
89  % que representan a cada categoria:
90  disp('************ Creando el clasificador de categorias... ************')
91  categoryClassifier = trainImageCategoryClassifier(trainingSets, bag);
92
93  %% Ahora evaluamos el clasificador
94
95  % Primero con el conjunto de entrenamiento
96  disp('********* Evaluando con el conjunto de entrenamiento... *********')
97  evaluate(categoryClassifier, trainingSets);
98
99  % Ahora con el conjunto de validacion
100 disp('********** Evaluando con el conjunto de validacion... **********')
101 confMatrix = evaluate(categoryClassifier, validationSets);
102 mean(diag(confMatrix));      % Compute average accuracy
103
104  %% Probamos con una imagen suelta
105 disp('************** Probando con una imagen suelta... **************')
106 img = imread(fullfile(rootFolder, 'vallas', 'vallas1.avi_frame285.jpg'));
107 [labelIdx, scores] = predict(categoryClassifier, img);
108
109  % Display the string label
110 categoryClassifier.Labels(labelIdx)
111
112  %% Paramos el contador
113 tiempo = toc;
114 fprintf('Tiempo transcurrido: %f\n',(tiempo/60));
```

B.2. ExtractFramesFromAVI.m

Este programa es el utilizado para extraer frames de los vídeos clasificados en diferentes directorios dependiendo de su categoría.

```
% Defino el directorio principal
rootFolder = fullfile('Youtube');

% Obtengo un listado de todos sus archivos
categorias = dir(rootFolder);
[num_categorias,y] = size(categorias);  % Obtengo el numero de categorias

% Itero sobre las categorias
for n = 3 : num_categorias
    % Obtengo el numero de videos dentro de esa categoria
    cat = fullfile(rootFolder, categorias(n).name);
    videos = dir(cat);
    [num_videos,y] = size(videos);

    % Itero sobre los videos
    for m = 3 : num_videos
        % Obtengo el nombre del video
        videoDir = fullfile(cat, videos(m).name);

        % Lo leo y saco el numero de frames que tiene
        video = VideoReader(videoDir);
        nFrames = video.NumberOfFrames;

        % Dependiendo del numero de frames del video, calculo una tabla
        % de numeros aleatorios que representaran un % de los frames
        % del video.
        tablaFrames = RandSinRepeticion(1,nFrames,nFrames*0.9);
        [x, framesElegidos] = size(tablaFrames);

        % Itero sobre los frames seleccionados aleatoriamente
        for f = 1 : framesElegidos
            % Obtengo el frame y lo guardo en JPG
            imageFrame = read(video,tablaFrames(1,f));
            imwrite(imageFrame, sprintf('Youtube/%s/%s_frame%d.jpg',
            categorias(n).name, videos(m).name, tablaFrames(1,f)));
        end
    end
end
```

B.3. RandSinRepeticion.m

Este script es utilizado para generar un rango de números aleatorios que son utilizados para la extracción de frames en el script anterior.

```
% Devuelve K valores aleatorios entre imin e imax

function [m] = RandSinRepeticion(imin,imax,K)

% Compruebo que el rango introducido este bien
if (imax-imin < K)
    fprintf(' Error:excede el rango\n');
    m = NaN;
    return
end

n = 0; % contador de numeros aleatorios
m = imin-1;

% Itero obteniendo los numeros y guardandolos en m si no estan repetidos
while (n < K)
    a = randi([imin,imax],1);

    if ((a == m) == 0)
        m = [m, a];
        n = n+1;
    end
end

m = m(:,2:end);
```

B.4. ResizeImages.m

Código utilizado para modificar la resolución de los frames de los datasets.

```
% Defino el directorio principal
rootFolder = fullfile('Nieve_frames_dataset_YouTube');

% Obtengo un listado de todos sus archivos
categorias = dir(rootFolder);
[num_categorias,y] = size(categorias);  % Obtengo el numero de categorias

% Itero sobre las categorias
for n = 3 : num_categorias

    % Obtengo el numero de frames dentro de esa categoria
    cat = fullfile(rootFolder, categorias(n).name);
    frames = dir(cat);
    [num_frames,y] = size(frames);

    % Itero sobre los frames
    for m = 3 : num_frames
        % Obtengo el nombre del frame actual
        frameName = fullfile(cat, frames(m).name);
        img = imread(frameName);

        %if (size(img,1) ~= 480)
            img2 = imresize(img, [360,640]);
            imwrite(img2, sprintf('Redimensionadas/%s/%s',
            categorias(n).name, frames(m).name));
        %end
    end
end
```

B.5. ExtractorHOG.m

Código que implementa la extracción de descriptores HOG.

```
function [features, featureMetrics] = ExtractorHOG( img )

% Obtenemos los descriptores de HOG de la imagen
features = extractHOGFeatures(img);

```

```
6  % Usamos la varianza para escoger las features mas fuertes
7  featureMetrics = var(features,[],2);
8
9  end
```

B.6. ExtractorHOF.m

Código que implementa la extracción de descriptores HOF.

```
1  function [features, featureMetrics] = ExtractorHOF( img )
2
3  alpha = 10;
4  iterations = 3;
5  calcularHOF = 0;
6  hsize = [5 5];
7  sigma = 1;
8
9  % Cogemos el valor de la variable global
10 pareja = getGlobal;
11
12 % Comprobamos si esta vacia
13 if (isempty(pareja{1,1}))
14     pareja{1,1} = img;
15     setGlobal(pareja);
16
17 % Segunda iteracion (solo 1 imagen guardada)
18 elseif (isempty(pareja{2,1}))
19     pareja{2,1} = img;
20     setGlobal(pareja);
21     calcularHOF = 1;
22
23 % Actualizamos el frame actual y borramos el viejo
24 else
25     pareja{1,1} = pareja{2,1};
26     pareja{2,1} = img;
27     setGlobal(pareja);
28     calcularHOF = 1;
29 end
30
31 if (calcularHOF == 1)
32     % Frame anterior
33     I = rgb2gray(pareja{1,1});
34     h = fspecial('gaussian',hsize,sigma);
```

```
35      PreviousFrame = imfilter(I,h,'replicate');
36
37      % Se reduce el tama\~no de la imagen anterior
38      PreviousFrame = impyramid(PreviousFrame,'reduce');
39      PreviousFrame = impyramid(PreviousFrame,'reduce');
40      [height, width] = size(PreviousFrame);
41
42      % Frame actual
43      I = rgb2gray(pareja{2,1});
44      CurrentFrame = imfilter(I,h,'replicate');
45
46      % Se reduce el tama\~no de la imagen actual
47      CurrentFrame = impyramid(CurrentFrame,'reduce');
48      CurrentFrame = impyramid(CurrentFrame,'reduce');
49
50      % Obtenemos los descriptores HOF de la imagen
51      [TempNormal] = Opticalflow(CurrentFrame,PreviousFrame,alpha,iterations,
52                                 height,width);
53      features = TempNormal;
54
55      % Usamos la varianza para escoger las features mas fuertes
56      featureMetrics = var(features,[],2);
57  else
58      features = zeros(90,160);
59      featureMetrics = var(features,[],2);
60  end
```

B.7. ExtractorMBHx.m

Código que implementa la extracción de descriptores MBHx.

```
1  function [features, featureMetrics] = ExtractorMBHx( img )
2
3  alpha = 10;
4  iterations = 3;
5  calcularHOF = 0;
6  hsize = [5 5];
7  sigma = 1;
8
9   % Cogemos el valor de la variable global
10 pareja = getGlobal;
11
12  % Comprobamos si esta vacia
13 if (isempty(pareja{1,1}))
14     pareja{1,1} = img;
15     setGlobal(pareja);
16
17  % Segunda iteracion (solo 1 imagen guardada)
18 elseif (isempty(pareja{2,1}))
19     pareja{2,1} = img;
20     setGlobal(pareja);
21     calcularHOF = 1;
22
23  % Actualizamos el frame actual y borramos el viejo
24 else
25     pareja{1,1} = pareja{2,1};
26     pareja{2,1} = img;
27     setGlobal(pareja);
28     calcularHOF = 1;
29 end
30
31 if (calcularHOF == 1)
32      % Frame anterior
33     I = rgb2gray(pareja{1,1});
34     h = fspecial('gaussian',hsize,sigma);
35     PreviousFrame = imfilter(I,h,'replicate');
36
37     % Se reduce el tama\~no de la imagen anterior
38     PreviousFrame = impyramid(PreviousFrame,'reduce');
39     PreviousFrame = impyramid(PreviousFrame,'reduce');
40     [height, width] = size(PreviousFrame);
41
```

```
42      % Frame actual
43      I = rgb2gray(pareja{2,1});
44      CurrentFrame = imfilter(I,h,'replicate');
45
46      % Se reduce el tama\~no de la imagen actual
47      CurrentFrame = impyramid(CurrentFrame,'reduce');
48      CurrentFrame = impyramid(CurrentFrame,'reduce');
49
50      % Obtenemos los descriptores HOF de la imagen
51      [TempU] = Opticalflow(CurrentFrame,PreviousFrame,alpha,iterations,
52                            height,width);
53
54      % Para MBHx:
55      features = extractHOGFeatures(TempU);
56
57      % Usamos la varianza para escoger las features mas fuertes
58      featureMetrics = var(features,[],2);
59  else
60      features = zeros(1,6840);
61      featureMetrics = var(features,[],2);
62  end
```

B.8. ExtractorMBHy.m

Código que implementa la extracción de descriptores MBHy.

```
1   function [features, featureMetrics] = ExtractorMBHy( img )
2
3   alpha = 10;
4   iterations = 3;
5   calcularHOF = 0;
6   hsize = [5 5];
7   sigma = 1;
8
9   % Cogemos el valor de la variable global
10  pareja = getGlobal;
11
12  % Comprobamos si esta vacia
13  if (isempty(pareja{1,1}))
14      pareja{1,1} = img;
15      setGlobal(pareja);
16
17  % Segunda iteracion (solo 1 imagen guardada)
```

```
18 elseif (isempty(pareja{2,1}))
19     pareja{2,1} = img;
20     setGlobal(pareja);
21     calcularHOF = 1;
22
23  % Actualizamos el frame actual y borramos el viejo
24 else
25     pareja{1,1} = pareja{2,1};
26     pareja{2,1} = img;
27     setGlobal(pareja);
28     calcularHOF = 1;
29 end
30
31 if (calcularHOF == 1)
32     % Frame anterior
33     I = rgb2gray(pareja{1,1});
34     h = fspecial('gaussian',hsize,sigma);
35     PreviousFrame = imfilter(I,h,'replicate');
36
37     % Se reduce el tama\~no de la imagen anterior
38     PreviousFrame = impyramid(PreviousFrame,'reduce');
39     PreviousFrame = impyramid(PreviousFrame,'reduce');
40     [height, width] = size(PreviousFrame);
41
42     % Frame actual
43     I = rgb2gray(pareja{2,1});
44     CurrentFrame = imfilter(I,h,'replicate');
45
46     % Se reduce el tama\~no de la imagen actual
47     CurrentFrame = impyramid(CurrentFrame,'reduce');
48     CurrentFrame = impyramid(CurrentFrame,'reduce');
49
50     % Obtenemos los descriptores HOF de la imagen
51     [TempV] = Opticalflow(CurrentFrame,PreviousFrame,alpha,iterations,
52                           height,width);
53
54     % Para MBHy:
55     features = extractHOGFeatures(TempV);
56
57     % Usamos la varianza para escoger las features mas fuertes
58     featureMetrics = var(features,[],2);
59 else
60     features = zeros(1,6840);
61     featureMetrics = var(features,[],2);
62 end
```

B.9. ExtractorSURF.m

Código que implementa la extracción de descriptores SURF.

```
function [features, featureMetrics] = ExtractorSURF( img )

I = rgb2gray(img);
points = detectSURFFeatures(I);
[features] = extractFeatures(I, points);

featureMetrics = var(features,[],2);

end
```

B.10. Opticalflow.m

Este programa es el encargado de calcular el flujo óptico que se utiliza en HOF, MBHx y MBHy.

```
function [TempU TempV TempNormal] =
Opticalflow( CurrentFrame, PreviousFrame, alpha, iterations,height,width)


TempU=zeros(height,width);
TempV=zeros(height,width);
Window=[1/12 1/6 1/12;1/6 0 1/6;1/12 1/6 1/12];

TempEx = conv2(double(CurrentFrame),double(0.25*[-1,1;-1,1]),'same') +
conv2(double(PreviousFrame),double(0.25*[-1  1; -1  1]),'same');

TempEy= conv2(double(CurrentFrame), double(0.25*[-1,-1;1,1]),'same') +
conv2(double(PreviousFrame),double(0.25*[-1  -1; 1  1]), 'same');

TempEt= conv2(double(CurrentFrame), double(0.25*ones(2)),'same') +
conv2(double(PreviousFrame), double(-0.25*ones(2)),'same');

TempNormal=TempEt./sqrt(TempEy.^2+TempEx.^2);
TempNormal(isnan(TempNormal))=0;
TempNormal(isinf(TempNormal))=0;
% size(TempEx);
for i=iterations
    ubar=conv2(double(TempU),Window,'same');
```

```
24      vbar=conv2(double(TempV),Window,'same');
25
26      % Compute flow vectors constrained by its local average and the optical
27      % flow constraints
28      TempU= ubar - ( TempEx.* ( ( TempEx.* ubar ) + ( TempEy.* vbar ) +
29      TempEt ) ) ./ ( alpha^2 + TempEx.^2 + TempEy.^2);
30      TempV= vbar - ( TempEy.* ( ( TempEx.* ubar ) + ( TempEy.* vbar ) +
31      TempEt ) ) ./ ( alpha^2 + TempEx.^2 + TempEy.^2);
32  end
```

B.11. getGlobal.m

Función utilizada para guardar y devolver una variable global utilizada en el cálculo de descriptores con flujo óptico.

```
1  function par = getGlobal
2      global parejasImg;
3      par = parejasImg;
4  end
```

B.12. setGlobal.m

Función utilizada para cambiar el valor de una variable global utilizada en el cálculo de descriptores con flujo óptico.

```
1  function setGlobal( tablaImgs )
2      global parejasImg;
3      parejasImg = tablaImgs;
4  end
```

B.13. graficas.m

Script encargado de generar todas las gráficas del proyecto.

```
%%%%%%%%%%%%%%%%%%%%%
%   MI DATASET      %
%%%%%%%%%%%%%%%%%%%%%

%%%%HOF %%%%
y = [89 74 ; 49 29 ; 61 30 ; 55 26 ; 87 65];  % [entrenamiento validacion;]
figure
bar(y)
set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
'personas' 'sombras'})

x = [0.85 1.15];
for w=1:size(y,1)
    y_aux = y(w,:);      % Cojo una fila de y
    for i=1:numel(y_aux)
        text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
        'HorizontalAlignment','center','VerticalAlignment','bottom')
    end
    x = x+1;
end

title('Mi Dataset - HOF')
legend('Entrenamiento','Validacion','Location','southoutside')
ylim([0 100]);

%%%%HOG %%%%
y = [79 59 ; 76 45 ; 74 41 ; 95 70 ; 84 65];  % [entrenamiento validacion;]
figure
bar(y)
set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
'personas' 'sombras'})

x = [0.85 1.15];
for w=1:size(y,1)
    y_aux = y(w,:);      % Cojo una fila de y
    for i=1:numel(y_aux)
        text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
        'HorizontalAlignment','center','VerticalAlignment','bottom')
    end
    x = x+1;
end
```

```
42
43 title('Mi Dataset - HOG')
44 legend('Entrenamiento','Validacion','Location','southoutside')
45
46 %%%%MBHx %%%%
47 y = [92 82 ; 80 48 ; 84 53 ; 74 31 ; 96 73];  % [entrenamiento validacion;]
48 figure
49 bar(y)
50 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
51 'personas' 'sombras'})
52
53 x = [0.85 1.15];
54 for w=1:size(y,1)
55     y_aux = y(w,:);      % Cojo una fila de y
56     for i=1:numel(y_aux)
57         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
58         'HorizontalAlignment','center','VerticalAlignment','bottom')
59     end
60     x = x+1;
61 end
62
63 title('Mi Dataset - MBHx')
64 legend('Entrenamiento','Validacion','Location','southoutside')
65
66 %%%%MBHy %%%%
67 y = [89 80 ; 79 42 ; 81 44 ; 91 57 ; 90 78];  % [entrenamiento validacion;]
68 figure
69 bar(y)
70 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
71 'personas' 'sombras'})
72
73 x = [0.85 1.15];
74 for w=1:size(y,1)
75     y_aux = y(w,:);      % Cojo una fila de y
76     for i=1:numel(y_aux)
77         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
78         'HorizontalAlignment','center','VerticalAlignment','bottom')
79     end
80     x = x+1;
81 end
82
83 title('Mi Dataset - MBHy')
84 legend('Entrenamiento','Validacion','Location','southoutside')
85
86 %%%%SURF %%%%
87 % [entrenamiento validacion;]
88 y = [100 97 ; 95 80 ; 98 90 ; 98 92 ; 97 92];
```

```
89  figure
90  bar(y)
91  set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
92  'personas' 'sombras'})
93
94  x = [0.85 1.15];
95  for w=1:size(y,1)
96      y_aux = y(w,:);      % Cojo una fila de y
97      for i=1:numel(y_aux)
98          text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
99          'HorizontalAlignment','center','VerticalAlignment','bottom')
100     end
101     x = x+1;
102 end
103
104 title('Mi Dataset - SURF')
105 legend('Entrenamiento','Validacion','Location','southoutside')
106
107 %%%%GENERAL %%%%
108 % [precision_media_entrenamiento precision_media_validacion;]
109 y = [68 45 ; 82 56 ; 85 58 ; 86 60 ; 98 90];
110 figure
111 bar(y)
112 set(gca, 'XTick',1:5, 'XTickLabel',{'HOF' 'HOG' 'MBHx' 'MBHy' 'SURF'})
113
114 x = [0.85 1.15];
115 for w=1:size(y,1)
116     y_aux = y(w,:);      % Cojo una fila de y
117     for i=1:numel(y_aux)
118         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
119         'HorizontalAlignment','center','VerticalAlignment','bottom')
120     end
121     x = x+1;
122 end
123
124 title('Mi Dataset - Precision media general')
125 legend('Entrenamiento','Validacion','Location','southoutside')
126
127 %%%%%%%%%%%%%%%%%%%%%%
128 %   DATASET YOUTUBE %
129 %%%%%%%%%%%%%%%%%%%%%%
130
131 %%%%HOF %%%%
132 y = [84 64 ; 59 42 ; 65 40 ; 73 44 ; 90 81];   % [entrenamiento validacion;]
133 figure
134 bar(y)
135 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
```

```
136 'personas' 'sombras'})
137
138 x = [0.85 1.15];
139 for w=1:size(y,1)
140     y_aux = y(w,:);      % Cojo una fila de y
141     for i=1:numel(y_aux)
142         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
143         'HorizontalAlignment','center','VerticalAlignment','bottom')
144     end
145     x = x+1;
146 end
147
148 title('Dataset YouTube - HOF')
149 legend('Entrenamiento','Validacion','Location','southoutside')
150 ylim([0 100]);
151
152 %%%%HOG %%%%
153 y = [92 78 ; 92 77 ; 92 78 ; 92 78 ; 92 83];  % [entrenamiento validacion;]
154 figure
155 bar(y)
156 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
157 'personas' 'sombras'})
158
159 x = [0.85 1.15];
160 for w=1:size(y,1)
161     y_aux = y(w,:);      % Cojo una fila de y
162     for i=1:numel(y_aux)
163         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
164         'HorizontalAlignment','center','VerticalAlignment','bottom')
165     end
166     x = x+1;
167 end
168
169 title('Dataset YouTube - HOG')
170 legend('Entrenamiento','Validacion','Location','southoutside')
171
172 %%%%MBHx %%%%
173 y = [92 62 ; 94 80 ; 92 81 ; 95 75 ; 98 96];  % [entrenamiento validacion;]
174 figure
175 bar(y)
176 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
177 'personas' 'sombras'})
178
179 x = [0.85 1.15];
180 for w=1:size(y,1)
181     y_aux = y(w,:);      % Cojo una fila de y
182     for i=1:numel(y_aux)
```

```
183         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
184         'HorizontalAlignment','center','VerticalAlignment','bottom')
185     end
186     x = x+1;
187 end
188
189 title('Dataset YouTube - MBHx')
190 legend('Entrenamiento','Validacion','Location','southoutside')
191
192 %%%%MBHy %%%%
193 % [entrenamiento validacion;]
194 y = [87 66 ; 92 69 ; 94 70 ; 97 86 ; 100 89];
195 figure
196 bar(y)
197 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
198 'personas' 'sombras'})
199
200 x = [0.85 1.15];
201 for w=1:size(y,1)
202     y_aux = y(w,:);      % Cojo una fila de y
203     for i=1:numel(y_aux)
204         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
205         'HorizontalAlignment','center','VerticalAlignment','bottom')
206     end
207     x = x+1;
208 end
209
210 title('Dataset YouTube - MBHy')
211 legend('Entrenamiento','Validacion','Location','southoutside')
212
213 %%%%SURF %%%%
214 % [entrenamiento validacion;]
215 y = [95 87 ; 100 90 ; 95 87 ; 100 99 ; 100 98];
216 figure
217 bar(y)
218 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
219 'personas' 'sombras'})
220
221 x = [0.85 1.15];
222 for w=1:size(y,1)
223     y_aux = y(w,:);      % Cojo una fila de y
224     for i=1:numel(y_aux)
225         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
226         'HorizontalAlignment','center','VerticalAlignment','bottom')
227     end
228     x = x+1;
229 end
```

```
230
231 title('Dataset YouTube - SURF')
232 legend('Entrenamiento','Validacion','Location','southoutside')
233
234 %%%%GENERAL %%%%
235 % [precision_media_entrenamiento precision_media_validacion;]
236 y = [74 54 ; 92 79 ; 94 79 ; 94 76 ; 98 92];
237 figure
238 bar(y)
239 set(gca, 'XTick',1:5, 'XTickLabel',{'HOF' 'HOG' 'MBHx' 'MBHy' 'SURF'})
240
241 x = [0.85 1.15];
242 for w=1:size(y,1)
243     y_aux = y(w,:);      % Cojo una fila de y
244     for i=1:numel(y_aux)
245         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
246         'HorizontalAlignment','center','VerticalAlignment','bottom')
247     end
248     x = x+1;
249 end
250
251 title('Dataset YouTube - Precision media general')
252 legend('Entrenamiento','Validacion','Location','southoutside')
253
254 %%%%%%%%%%%%%%%%%%%%%
255 %   DATASET MEZCLADO %
256 %%%%%%%%%%%%%%%%%%%%%
257
258 %%%%HOF %%%%
259 y = [74 47 ; 65 24 ; 63 24 ; 76 47 ; 80 67];  % [entrenamiento validacion;]
260 figure
261 bar(y)
262 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
263 'personas' 'sombras'})
264
265 x = [0.85 1.15];
266 for w=1:size(y,1)
267     y_aux = y(w,:);      % Cojo una fila de y
268     for i=1:numel(y_aux)
269         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
270         'HorizontalAlignment','center','VerticalAlignment','bottom')
271     end
272     x = x+1;
273 end
274
275 title('Dataset Mezclado - HOF')
276 legend('Entrenamiento','Validacion','Location','southoutside')
```

```
277 ylim([0 100]);
278
279 %%%%HOG %%%%
280 y = [86 73 ; 80 62 ; 86 67 ; 91 64 ; 94 86];  % [entrenamiento validacion;]
281 figure
282 bar(y)
283 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
284 'personas' 'sombras'})
285
286 x = [0.85 1.15];
287 for w=1:size(y,1)
288     y_aux = y(w,:);      % Cojo una fila de y
289     for i=1:numel(y_aux)
290         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
291         'HorizontalAlignment','center','VerticalAlignment','bottom')
292     end
293     x = x+1;
294 end
295
296 title('Dataset Mezclado - HOG')
297 legend('Entrenamiento','Validacion','Location','southoutside')
298
299 %%%%MBHx %%%%
300 y = [96 71 ; 80 61 ; 85 57 ; 94 65 ; 98 85];  % [entrenamiento validacion;]
301 figure
302 bar(y)
303 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
304 'personas' 'sombras'})
305
306 x = [0.85 1.15];
307 for w=1:size(y,1)
308     y_aux = y(w,:);      % Cojo una fila de y
309     for i=1:numel(y_aux)
310         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
311         'HorizontalAlignment','center','VerticalAlignment','bottom')
312     end
313     x = x+1;
314 end
315
316 title('Dataset Mezclado - MBHx')
317 legend('Entrenamiento','Validacion','Location','southoutside')
318
319 %%%%MBHy %%%%
320 y = [96 69 ; 84 56 ; 95 56 ; 94 61 ; 95 72];  % [entrenamiento validacion;]
321 figure
322 bar(y)
323 set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
```

```
'personas' 'sombras'})

x = [0.85 1.15];
for w=1:size(y,1)
    y_aux = y(w,:);      % Cojo una fila de y
    for i=1:numel(y_aux)
        text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
        'HorizontalAlignment','center','VerticalAlignment','bottom')
    end
    x = x+1;
end

title('Dataset Mezclado - MBHy')
legend('Entrenamiento','Validacion','Location','southoutside')

%%%%SURF %%%%
% [entrenamiento validacion;]
y = [94 78 ; 96 79 ; 92 78 ; 98 93 ; 100 93];
figure
bar(y)
set(gca, 'XTick',1:5, 'XTickLabel',{'balizas' 'canones' 'vallas'
'personas' 'sombras'})

x = [0.85 1.15];
for w=1:size(y,1)
    y_aux = y(w,:);      % Cojo una fila de y
    for i=1:numel(y_aux)
        text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
        'HorizontalAlignment','center','VerticalAlignment','bottom')
    end
    x = x+1;
end

title('Dataset Mezclado - SURF')
legend('Entrenamiento','Validacion','Location','southoutside')

%%%%GENERAL %%%%
% [precision_media_entrenamiento precision_media_validacion;]
y = [71 42 ; 87 70 ; 91 68 ; 93 63 ; 96 84];
figure
bar(y);
set(gca, 'XTick',1:5, 'XTickLabel',{'HOF' 'HOG' 'MBHx' 'MBHy' 'SURF'})

x = [0.85 1.15];
for w=1:size(y,1)
    y_aux = y(w,:);      % Cojo una fila de y
    for i=1:numel(y_aux)
```

```
371         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
372         'HorizontalAlignment','center','VerticalAlignment','bottom')
373     end
374     x = x+1;
375 end
376
377 title('Dataset Mezclado - Precision media general')
378 legend('Entrenamiento','Validacion','Location','southoutside')
379
380 %%%%%%%%%%%%%%%%%%%%%%%%
381 %   GRAFICAS GENERALES   %
382 %%%%%%%%%%%%%%%%%%%%%%%%
383
384 %%%%COMPARACION DATASETS %%%%
385 % [precision_media_validacion;]
386 y = [45 56 58 60 90 ; 54 79 79 76 92 ; 42 70 68 63 84];
387 figure
388 bar(y)
389 set(gca, 'XTick',1:3, 'XTickLabel',{'Mi Dataset' 'Dataset YouTube'
390 'Dataset Mezclado'})
391
392 x = [0.68 0.85 1 1.15 1.3];
393 for w=1:size(y,1)
394     y_aux = y(w,:);      % Cojo una fila de y
395     for i=1:numel(y_aux)
396         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
397         'HorizontalAlignment','center','VerticalAlignment','bottom')
398     end
399     x = x+1;
400 end
401
402 title('Comparacion Datasets')
403 legend('HOF','HOG','MBHx','MBHy','SURF','Location','eastoutside')
404
405 %%%%MI DATASET - COMPARACION OBJETOS %%%%
406 y = [ 74 59 82 80 97 ; 29 45 48 42 80 ; 30 41 53 44 90 ; 26 70 31 57 92;
407       65 65 73 78 92];     % [validacion de cada objeto;]
408 figure
409 bar(y)
410 set(gca, 'XTick',1:5, 'XTickLabel',{'Balizas' 'Canones' 'Vallas'
411 'Personas' 'Sombras'})
412
413 x = [0.68 0.85 1 1.15 1.3];
414 for w=1:size(y,1)
415     y_aux = y(w,:);      % Cojo una fila de y
416     for i=1:numel(y_aux)
417         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
```

```
418         'HorizontalAlignment','center','VerticalAlignment','bottom')
419     end
420     x = x+1;
421 end
422
423 title('Mi Dataset - Comparacion Objetos')
424 legend('HOF','HOG','MBHx','MBHy','SURF','Location','eastoutside')
425
426 %%%%DATASET YOUTUBE - COMPARACION OBJETOS %%%%
427 y = [ 64 78 62 66 87 ; 42 77 80 69 90 ; 40 78 81 70 87 ; 44 78 75 86 99 ;
428      81 83 96 89 98 ];     % [validacion de cada objeto;]
429 figure
430 bar(y)
431 set(gca, 'XTick',1:5, 'XTickLabel',{'Balizas' 'Canones' 'Vallas'
432 'Personas' 'Sombras'})
433
434 x = [0.68 0.85 1 1.15 1.3];
435 for w=1:size(y,1)
436     y_aux = y(w,:);      % Cojo una fila de y
437     for i=1:numel(y_aux)
438         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
439         'HorizontalAlignment','center','VerticalAlignment','bottom')
440     end
441     x = x+1;
442 end
443
444 title('Dataset YouTube - Comparacion Objetos')
445 legend('HOF','HOG','MBHx','MBHy','SURF','Location','eastoutside')
446
447 %%%%DATASET MEZCLADO - COMPARACION OBJETOS %%%%
448 y = [ 47 73 71 69 78 ; 24 62 61 56 79 ; 24 67 57 56 78 ; 47 64 65 61 93 ;
449      67 86 85 72 93 ];     % [validacion de cada objeto;]
450 figure
451 bar(y)
452 set(gca, 'XTick',1:5, 'XTickLabel',{'Balizas' 'Canones' 'Vallas'
453 'Personas' 'Sombras'})
454
455 x = [0.68 0.85 1 1.15 1.3];
456 for w=1:size(y,1)
457     y_aux = y(w,:);      % Cojo una fila de y
458     for i=1:numel(y_aux)
459         text(x(1,i),y_aux(1,i),num2str(y_aux(1,i),'%0.0f'),
460         'HorizontalAlignment','center','VerticalAlignment','bottom')
461     end
462     x = x+1;
463 end
464
```

```
465 title('Dataset Mezclado - Comparacion Objetos')
466 legend('HOF','HOG','MBHx','MBHy','SURF','Location','eastoutside')
```

B.14. tiemposDescriptores.m

Programa utilizado para la obtención del tiempo de cálculo de cada descriptor dada una imagen.

```
1  img = imread('valla6.mp4_frame47.jpg');
2  tipoDescrip = 5;
3
4  switch tipoDescrip
5      case 1
6          tic     % Iniciamos el contador de tiempo
7          [features, featureMetrics] = ExtractorHOG(img);
8          tiempo = toc;    % Paramos el contador
9
10     case 2
11         parejasImg = cell(2,1);
12         setGlobal(parejasImg);
13         ExtractorHOF(img);
14         img2 = imread('valla6.mp4_frame48.jpg');
15         tic     % Iniciamos el contador de tiempo
16         [features, featureMetrics] = ExtractorHOF(img2);
17         tiempo = toc;    % Paramos el contador
18
19     case 3
20         parejasImg = cell(2,1);
21         setGlobal(parejasImg);
22         ExtractorMBHx(img);
23         img2 = imread('valla6.mp4_frame48.jpg');
24         tic     % Iniciamos el contador de tiempo
25         [features, featureMetrics] = ExtractorMBHx(img2);
26         tiempo = toc;    % Paramos el contador
27
28     case 4
29         parejasImg = cell(2,1);
30         setGlobal(parejasImg);
31         ExtractorMBHy(img);
32         img2 = imread('valla6.mp4_frame48.jpg');
33         tic     % Iniciamos el contador de tiempo
34         [features, featureMetrics] = ExtractorMBHy(img2);
35         tiempo = toc;    % Paramos el contador
```

```
36
37      case 5
38          tic      % Iniciamos el contador de tiempo
39          [features, featureMetrics] = ExtractorSURF(img);
40          tiempo = toc;    % Paramos el contador
41
42      otherwise
43          disp('El descriptor elegido no existe.')
44  end
45
46
47  fprintf('Tiempo transcurrido: %f\n',(tiempo));
```

Apéndice C

Detalle del diagrama de Gantt

Tarea

Nombre	Fecha de inicio	Fecha de fin	Horas
Terminando de completar el dataset de youtube. Clasificando las imágenes.	27/12/15	27/12/15	3
Cambiando tamaño a todas las imágenes del dataset de youtube	9/01/16	9/01/16	2
Creación dataset mezclado	13/01/16	13/01/16	2
Matlab	24/03/15	29/12/15	13
Instalación de la versión 2014b (Linux)	24/03/15	25/04/15	8
Problemas memoria Matlab	12/05/15	12/05/15	1
He incrementado la memoria Swap en 1GB y la RAM hasta 4GB pero sigue sin funcionar.			
Instalación de la versión 2015a (Windows)	26/06/15	26/06/15	2
La implementación que necesitamos del CustomExtractor en la función BagOfFeatures no está en la versión 2014b. Busco, descargo e instalo la última versión para Windows.			
Problema longitud descriptores	29/12/15	29/12/15	2
Problemas con el tamaño de las imágenes del nuevo dataset. Deben ser todas del mismo tamaño o el bagoffeatures da error en la longitud de descriptores.			
Bag of Words	27/03/15	7/07/15	17
Ejecución Bag Of Words Matlab	27/03/15	27/03/15	2
Ejecución del ejemplo BoW de Matlab con imágenes	26/04/15	26/04/15	2
Montando BoW con videos	30/04/15	30/04/15	2
Extracción de frames de los videos de youtube y creación del BoW	2/05/15	2/05/15	3
Implementación del cálculo de descriptores HOG y HOF	28/06/15	28/06/15	4
Implementación del cálculo de descriptores MBHx y MBHy	1/07/15	1/07/15	2
Probando con descriptor LDOF, intentando ejecutarlo en Matlab	6/07/15	6/07/15	1
Arreglo del descriptor MBH	7/07/15	7/07/15	1
Estudio de códigos fuente	31/05/15	18/06/15	4
Código del DenseTrack.cpp	31/05/15	31/05/15	2

Tarea

Nombre	Fecha de inicio	Fecha de fin	Horas
Código RealtimeHogHofReleaseSource	15/06/15	18/06/15	2
Combinación de descriptores	27/07/15	28/11/15	25
Creando función SURF particular. Iniciando la combinación.	27/07/15	27/07/15	2
Comprobando tamaño del 'featureVector'	5/08/15	5/08/15	1
Comprobando tamaño del 'featureVector' para todos los tipos de descriptores.			
Aumentando la cantidad de imágenes	13/09/15	14/09/15	3
Aumentando la cantidad de imágenes para que los vectores de features tuvieran todos el mismo tamaño.			
Modificación de 'bagOfFeatures.m' - parte I	21/09/15	21/09/15	2
Modificación del código de matlab 'bagOfFeatures' para que devuelva el vocabulario.			
Modificación de 'bagOfFeatures.m' - parte II	19/10/15	24/10/15	5
Modificación de 'bagOfFeatures.m' para normalizar valores de los descriptores y poder calcularlos con más de un extractor.			
Combinando HOG y HOF	30/10/15	31/10/15	4
Trabajando con HOG y HOF intentando combinarlos. Extrayendo descriptores y concacatenándolos.			
Corrigiendo errores	1/11/15	10/11/15	4
Corrigiendo errores. Modificando función 'encodeSingleImage()' de 'bagOfFeatures.m'.			
Pruebas con y sin la función TRIM - parte I	16/11/15	16/11/15	2
Pruebas con y sin TRIM con un solo tipo de descriptor para ver su comportamiento.			
Pruebas con y sin la función TRIM - parte II	28/11/15	28/11/15	2
Pruebas con y sin TRIM con parejas de descriptores - HOG&HOF - MBHx&MBHy.			
Memoria	6/07/15	20/04/16	98
Borrador de la motivación	6/07/15	6/07/15	1
Objetivos del proyecto. Técnicas utilizadas (descripción de artículos). Boceto de Arquitectura.	22/07/15	22/07/15	4
Estructura de la memoria. Búsqueda de artículos relacionados.	26/07/15	26/07/15	2
Correcciones	6/08/15	6/08/15	2
Expansión de dispositivos de monitorización	7/08/15	7/08/15	1

Tarea

Nombre	Fecha de inicio	Fecha de fin	Horas
Ampliación del estado del arte	21/09/15	21/09/15	2
Ampliación de los resúmenes de articulos	28/09/15	29/09/15	3
Ejecutando descriptores por separado	10/12/15	10/12/15	1
Ejecutando descriptores por separado, para obtener datos finales para la memoria.			
Obtención de ejecuciones con 1 descriptor	18/12/15	18/12/15	3
Obtención de ejecuciones con 1 descriptor sin trim para comparar tiempos.			
Primeras pruebas con el dataset de youtube	27/12/15	27/12/15	2
Lanzando ejecuciones con el dataset de youtube	10/01/16	10/01/16	2
Lanzando ejecuciones con el dataset de youtube. Entrenando con mi dataset y validando con youtube.			
Probando ejecuciones con el dataset mezclado	13/01/16	13/01/16	1
Generación de gráficas	31/01/16	10/02/16	10
Detección de puntos en las imágenes	11/02/16	11/02/16	1
Detección de puntos en las imágenes para los resultados.			
Obteniendo imágenes con los descriptores superpuestos	17/02/16	17/02/16	3
Sacando resultados para la memoria. Obteniendo imágenes con los descriptores superpuestos.			
Describiendo cada descriptor	24/02/16	3/03/16	9
Pasando memoria a Latex	7/03/16	7/03/16	4
Volcando imágenes de las gráficas a la memoria	10/03/16	10/03/16	3
Descripción datasets	18/03/16	18/03/16	2
Correción errores	24/03/16	24/03/16	4
Construcción bibliografia	28/03/16	28/03/16	3
Resumen y prototipo	30/03/16	30/03/16	2
Prototipo software	2/04/16	2/04/16	4
Terminando anexos	4/04/16	4/04/16	1
Corrigiendo errores	7/04/16	11/04/16	5
Terminando anexo de gráficas	12/04/16	12/04/16	2
Tabla de tiempos y conclusiones	13/04/16	13/04/16	3
Corrigiendo errores	16/04/16	16/04/16	5
Creación anexo codigo fuente	17/04/16	17/04/16	2

Tarea

Nombre	Fecha de inicio	Fecha de fin	Horas
Diagrama Gantt	18/04/16	18/04/16	6
Correcciones	20/04/16	20/04/16	5

Bibliografía

[Dalal et al., 2006] Dalal, N., Triggs, B., and Schmid, C. (2006). Human Detection Using Oriented Histograms of Flow and Appearance. In Leonardis, A., Bischof, H., and Pinz, A., editors, *European Conference on Computer Vision (ECCV '06)*, volume 3952 of *Lecture Notes in Computer Science (LNCS)*, pages 428–441, Graz, Austria. Springer-Verlag. I, 11

[Faraki et al., 2014] Faraki, M., Palhang, M., and Sanderson, C. (2014). *CoRR*, abs/1406.2139. II, 10, 13

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). In *Determining Optical Flow*, volume 0281, pages 319–331. 20

[Jia et al., 2014] Jia, W., Hu, R. X., Lei, Y. K., Zhao, Y., and Gui, J. (2014). *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3):385–395. II, V, 13, 19

[Laptev et al., 2008] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. I, 11, 14

[Liu et al., 2009] Liu, J., Luo, J., and Shah, M. (2009). Recognizing realistic actions from videos in the wild. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1996–2003. I, 10, 12

[López-Sastre et al., 2013] López-Sastre, R. J., Renes-Olalla, J., Gil-Jiménez, P., Maldonado-Bascón, S., and Lafuente-Arroyo, S. (2013). *IEEE Transactions on Circuits and Systems for Video Technology*, 23(8):1358–1368. 11

[Matlab, 2016] Matlab (2016). Image classification with bag of visual words. `http://es.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html`. Accedido el 18-04-2016. V, 17

[Nanni et al., 2014] Nanni, L., Lumini, A., and Brahnam, S. (2014). *Journal of King Saud University - Science*, 26(2):89 – 100.

[Peng et al., 2014] Peng, X., Wang, L., Wang, X., and Qiao, Y. (2014). *CoRR*, abs/1405.4506.

[Uijlings et al., 2014] Uijlings, J., Duta, I. C., Sangineto, E., and Sebe, N. (2014). *International Journal of Multimedia Information Retrieval*, 4(1):33–44. II, 11, 14

[Wang et al., 2011] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2011). Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, Colorado Springs, United States.

[Wang et al., 2013] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). *International Journal of Computer Vision*, 103(1):60–79. I, 10, 13, 14

[Yuan et al., 2010] Yuan, C., Hu, W., Li, X., Maybank, S., and Luo, G. (2010). Human action recognition under log-euclidean riemannian metric. In *Proceedings of the 9th Asian Conference on Computer Vision - Volume Part I*, ACCV'09, pages 343–353, Berlin, Heidelberg. Springer-Verlag. I, 10, 12

[Zhang et al., 2006] Zhang, J., Marszałek, M., Lazebnik, S., and Schmid, C. (2006). *International Journal of Computer Vision*, 73(2):213–238. I, 11